MacTech Magazine
Vol. 13, No. 9 • September 1997

For Macintosh Programmers & Developers

MacTech®

INSIDE: MY FIRST YELLOW BOX APPLICATION

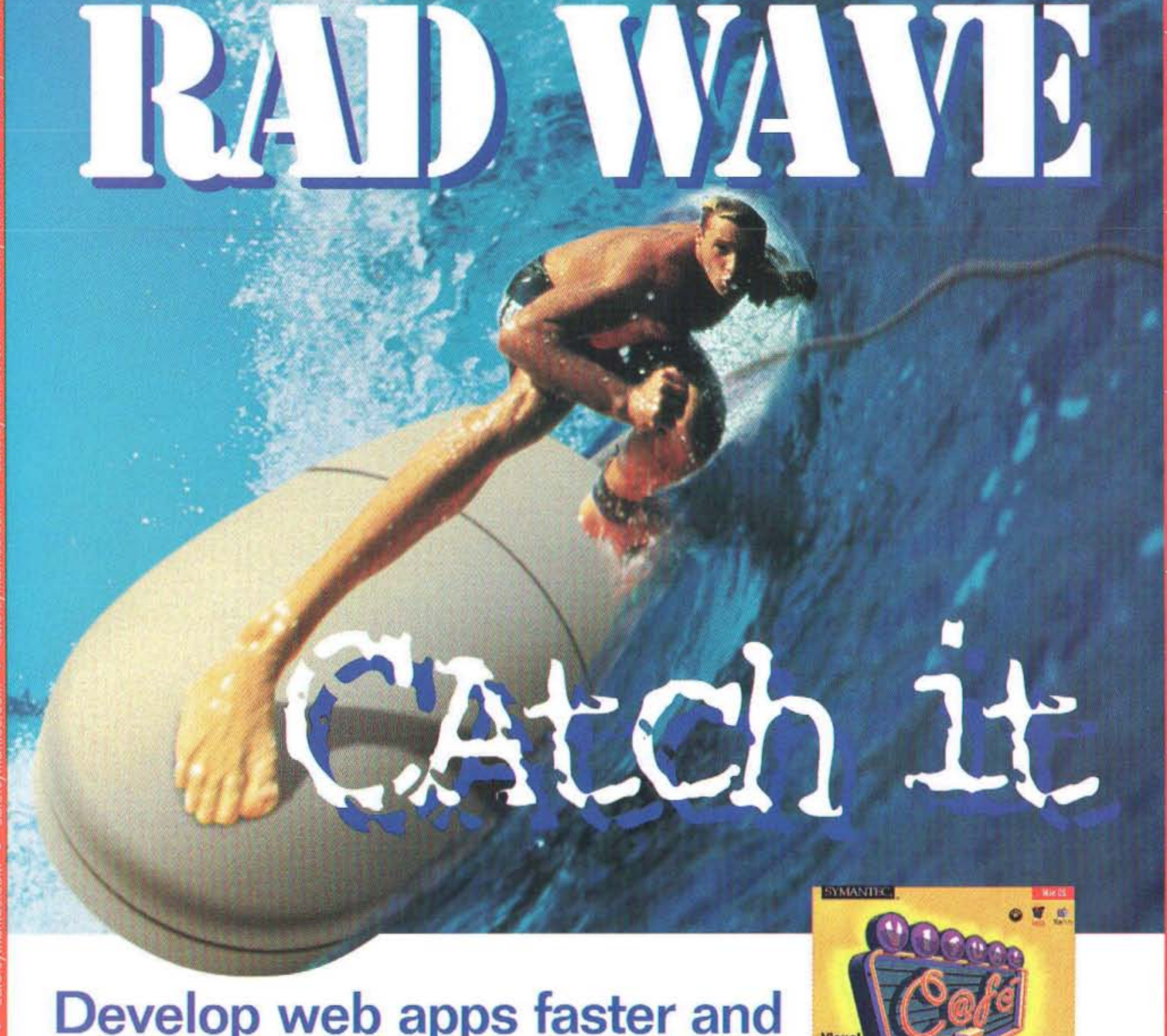
Includes Special develop Section by Apple Computer, Inc.

Mixing Java on the Mac

$5.85 US
$6.95 Canada
ISSN 1067-8360
Printed in U.S.A.

PLUS: The F Programming Language Tastes Like Java
Putting Java under PowerPlant • Contour Plotting in Java

cafe.symantec.com • cafe.symantec.com • cafe.symantec.com • cafe.symantec.com • cafe.symantec.com •



Develop web apps faster and easier with the only RAD Java™ tool for the Mac!

Symantec's Visual Café.™ The *only* Rapid Application Development (RAD) Java tool for the Mac. Guaranteed to let you knock out dynamic web apps at tsunami speed, with drag-and-drop ease. And best of all, your apps are ready to run on Rhapsody. So why use anything else? Call or visit our website today!

Mac OS

Visual Café™

Write in Java today, it runs in Rhapsody tomorrow!

Call for a FREE trial copy!
1-800-453-1059 ext 9H32
Or download from:
http://cafe.symantec.com

SYMANTEC.

cafe.symantec.com • cafe.symantec.com • cafe.symantec.com • cafe.symantec.com • cafe.symantec.com •

"Without a doubt, the Premiere Resource Editor for the Mac OS ... A wealth of time-saving tools."
– MacUser Magazine Eddy Awards

"A distinct improvement over Apple's ResEdit."
– MacTech Magazine

"Every Mac OS developer should own a copy of Resorcerer."
– Leonard Rosenthol, Aladdin Systems

"Without Resorcerer, our localization efforts would look like a Tower of Babel. Don't do product without it!"
– Greg Galanos, CEO and President, Metrowerks

"Resorcerer's data template system is amazing."
– Bill Goodman, author of Smaller Installer and Compact Pro

"Resorcerer Rocks! Buy it, you will NOT regret it."
*– Joe Zobkiw, author of **A Fragment of Your Imagination***

"Resorcerer will pay for itself many times over in saved time and effort."
– MacUser review

"The template that disassembles 'PICT's is awesome!"
– Bill Steinberg, author of Pyro! and PBTools

"Resorcerer proved indispensible in its own creation!"
– Doug McKenna, author of Resorcerer

Until Sept. 1
Only $128 !
when you order by credit card from our web site.

MACUSER EDITORS' CHOICE AWARD 1993 WINNER

Accelerated for Power Macintosh

Resorcerer® 2

Version 2.0

The Resource Editor for the Mac™ OS Wizard

ORDERING INFO

Requires System 7.0 or greater, 1.5MB RAM, CD-ROM

Standard price: $256 (decimal)
Website price: $128 - $256
(Educational, quantity, or other discounts available)

Includes: Electronic documentation
60-day Money-Back Guarantee
Domestic standard shipping

Payment: Check, PO's, or Visa/MC
Taxes: Colorado customers only

Extras (call, fax, or email us):
COD, FedEx, UPS Blue/Red,
International Shipping

MATHEMÆSTHETICS, INC.
PO Box 298
Boulder, CO 80306-0298 USA
Phone: (303) 440-0707
Fax: (303) 440-0504
resorcerer@mathemaesthetics.com

New in 2.0:

- **Very fast, HFS browser generically views any file's contents**
- **Extensibility for new Resorcerer Apprentices (CFM plug-ins)**
- **AppleScript Dictionary ('aete') Apprentice Editor**
- **Integrated support for 'Mcmd' menu commands**
- **TextTraits Editor**
- **Complete AIFF sound file disassembly template**
- **Big-, little-, and even mixed-endian template parsing**
- **HFS folder attribute editing**
- **Temporary memory always used everywhere if needed**
- **Automatically creates 'STR#'s by parsing C/C++ source**

- Easier, faster, more Mac-like, and more productive than ResEdit
- Safer memory-based, not disk-file-based, design and operation
- All file information and common commands in one easy-to-use window
- Compares resource files, and even **edits your data forks** as well
- Visible, accumulating, editable scrap
- Searches and opens/marks/selects resources by text content
- Makes global resource ID or type changes easily and safely
- Builds resource files from simple Rez-like scripts
- Most editors DeRez directly to the clipboard
- All graphic editors support screen-copying or partial screen-copying
- Hot-linking Value Converter for editing 32 bits in a dozen formats
- Its own 32-bit List Mgr can open and edit very large data structures
- Templates can pre- and post-process any arbitrary data structure
- Includes nearly 200 templates for common system resources
- TMPLs for Installer, MacApp, QT, Help, AppleEvent, OCE, GX, etc.
- Full integrated support for editing color dialogs and menus
- Try out balloons, 'ictb's, lists and popups, even create C source code
- Integrated single-window Hex/Code Editor, with patching, searching
- Editors for cursors, versions, pictures, bundles, and lots more
- Relied on by thousands of Macintosh developers around the world

To order by credit card, or to get the latest news, bug fixes, updates, and apprentices, visit our website...

www.mathemaesthetics.com

For Macintosh Programmers & Developers

MacTech® MAGAZINE

A publication of XPLAIN CORPORATION

How To Communicate With Us

In this electronic age, the art of communication has become both easier and more complicated. Is it any surprise that we prefer **e-mail?**

If you have any questions, feel free to call us at 805/494-9797 or fax us at 805/494-9798.

If you would like a subscription or need customer service, feel free to contact Developer Depot Customer Service at 800-MACDEV-1

DEPARTMENTS	E-Mail/URL
Orders, Circulation, & Customer Service	cust_service@devdepot.com
Press Releases	press_releases@mactech.com
Ad Sales	ad_sales@mactech.com
Editorial	editorial@mactech.com
Programmer's Challenge	prog_challenge@mactech.com
Online Support	online@mactech.com
Accounting	accounting@mactech.com
Marketing	marketing@mactech.com
General	info@mactech.com
Web Site (articles, info, URLs and more...)	http://www.mactech.com

MACTECH MAGAZINE

MacTech Magazine is grateful to the following individuals who contribute on a regular basis. We encourage others to share the technology.

We are dedicated to the distribution of useful programming information without regard to Apple's developer status.

For information on submitting articles, ask us for our ***writer's kit*** *which includes the terms and conditions upon which we publish articles.*

Editorial Board of Advisors
Scott T Boyd, Jordan J. Mattson, Jim Straus, and Jon Wiederspan

Editors
Publisher • Neil Ticktin
Editor-in-Chief • Eric Gundrum
Managing Editor • Jessica Courtney
Editorial Intern • Michelle Shin
Online Support • Nick "nick.c" DeMello

Contributing and Technical Editors
Components • Tantek Çelik, Microsoft Corporation
Internet • Carl de Cordova, Apple Computer, Inc.
Java • Will Iverson, Apple Computer, Inc.
MacDev-1™ • Rich Siegel, Bare Bones Software, Inc.
MacOS 8 • Steve Kiene, Mindvision
MagicCap/TeleScript • Richard Clark
Performance Programming • Jim Gochee, Connectix
Product Reviews • Ed Ringel
Rhapsody • Michael Rutman

Regular Columnists
Getting Started • Dave Mark
Programmer's Challenge • Bob Boonstra
From the Factory Floor • Dave Mark, Metrowerks
Tips & Tidbits • Steve Sisak
MacTech Online • Nick "nick.c" DeMello

XPLAIN CORPORATION

Chief Executive Officer • Neil Ticktin
Chief Operating Officer • Andrea J. Sniderman
Controller • Heather Principe
Advertising Executive • Ruth Subrin
Marketing Manager • Susan M. Worley
Network Administrator • Chris Barrus
Customer Relations • Erik Davidson, Stephanie Long, Lee Ann Pham, Susan Pomrantz
Accounting • Beth Kaelble, Jan Webber

Financial Services • Shin Financial Services
Art Direction/Production • InfoGraphix
Board of Advisors • Steven Geller, Blake Park, and Alan Carsrud

PRINTED WITH SOY INK

This publication is printed on paper with recycled content.

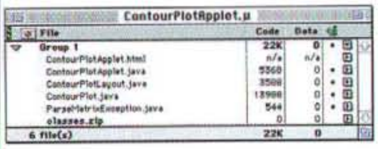
MacTech Magazine (ISSN: 1067-8360 / USPS: 010-227) is published monthly by Xplain Corporation, 850-P Hampshire Road, Westlake Village, CA 91361-2800. Voice: 805/494-9797, FAX: 805/494-9798. Domestic subscription rates are $47.00 per year. Canadian subscriptions are $59.00 per year. All other international subscriptions are $97.00 per year. Domestic source code disk subscriptions are $77 per year. All international disk subscriptions are $97.00 a year. Please remit in U.S. funds only. Periodical postage is paid at Thousand Oaks, CA and at additional mailing office.

POSTMASTER: Send address changes to **MacTech Magazine**, P.O. Box 5200, Westlake Village, CA 91359-5200.

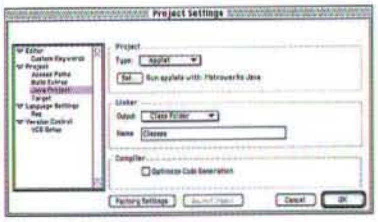
Contents

September 1997 • Volume 13, No.9

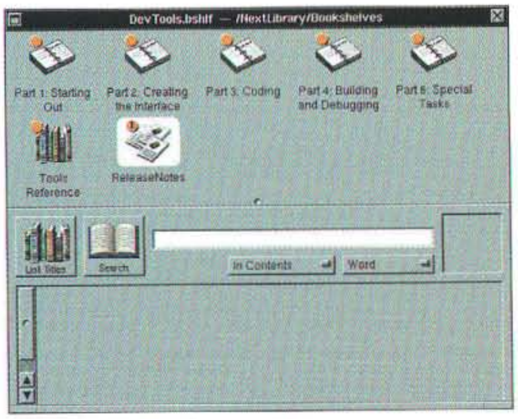
For Macintosh Programmers & Developers
MacTech® MAGAZINE



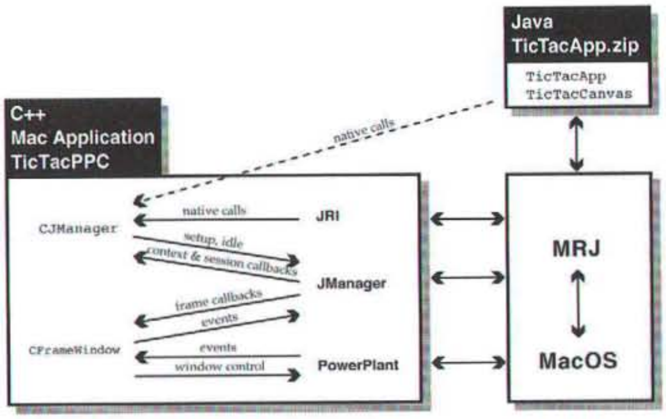
File	Code	Data
Group 1	22K	0
ContourPlotApplet.html	n/a	n/a
ContourPlotApplet.java	5360	0
ContourPlotLayout.java	3508	0
ContourPlot.java	13988	0
ParseMatrixException.java	544	0
classes.zip	0	0
6 file(s)	22K	0



Contour Plotting in Java, page 16



Getting Started, page 10



Putting Java Under PowerPlant, page 49

by Eric Gundrum

JAVA STILL BREWING

There has been more hype surrounding Sun's Java than anything else I remember about the computer industry. I am not entirely sure why Java has generated so much excitement. I suspect it has something to do with the promise of truly portable computing; that is, write code once and run it anywhere and everywhere. Unfortunately Java is still very far from realizing that promise. In fact, it is about as close to that promise as bad diner coffee is to a cup of home-brewed Sumatra.

Don't get me wrong, I want a product to do what Java promises just as much as everyone else. I would love to be able to sit down in a cafe with a public Java terminal, log into my web site, read mail, review my financial accounts, work on a few articles and write some code. Doing this from any computer, anywhere, and at any time would be wonderful. This capability would likely result in computers becoming more ubiquitous than telephones. However, Java is not up to it just yet.

A Well Designed Language

In recent weeks, I have been doing some research into Java. This language seems well designed when compared to the likes of C++ and the other common object-based languages. The Java designers seem to be well versed the various object-oriented languages and pulled the best principles and techniques from many of them.

I particularly like the appropriate mix of keywords and punctuation. People cite C++ as an obfuscating language partly because it tends to use punctuation for everything; just look at how many different meanings there are for a colon. C programmers complain about Pascal's use of keywords for everything. They get irritated by having to type "BEGIN" and "END" an infinite number of times when simple braces will do just fine.

Java's syntax looks a lot like C++ on the surface, but it includes some of the verbosity of Pascal where the extra verbiage can be a real benefit. Java's designers did not go overboard; they use C-style braces to define scope, so you won't get tired fingers from typing "BEGIN" and "END" all over the place. However, they added "implements," "extends," and other keywords to make the structure and meaning of the code clear from the text, rather than relying on your memory and the terse symbols of C++.

Another nice improvement is the absence of header files. Classes are declared in the same file in which they are implemented. The class declaration is included in the resulting binary object. Want to use a Java class in your applet? Just "import" the library file containing the class. The lack of header files, coupled with dynamic binding can eliminate many of the common problems coordinating the build process on multi-person projects. It also means fewer source files to maintain.

One of the most touted improvements is having memory management built into the language. No longer do we have to waste so much time tracing down rogue handles or other memory errors. Unfortunately, the implementations of garbage collection in most Java environments still leave a lot to be desired, but this is just an implementation issue. Sooner or later the Java vendors will provide more efficient garbage collection to gain performance better than their competitors.

Too Many Flavors

The biggest problem with multi-platform Java development is that there are too many flavors of Java and they don't quite work the same. This means that we must test our applications on all of the available Java platforms if we want to be sure it works for the multi-platform audience. Java programmers have reported to me that they can spend as much as 80% of their development time tracing and resolving platform-specific idiosyncrasies. Unfortunately, it looks like this problem will get much worse before it gets any better. There are many more implementations on the way, including those embedded in microchips.

What we need is an unambiguous specification for the language and libraries, and a certification process to ensure each implementation abides by the spec. For the moment, Sun seems too busy pumping the Java media wave to deal with this issue. (Sun's "100% Java" campaign is an example of this hype.) If this problem is not solved soon, it will likely be what stops the seeming endless momentum of Java.

Conclusions

Java is a nice evolutionary step beyond C++, but it is not the be-all and end-all of languages. Java incorporates features other languages have had for years, bringing these features into the mainstream, but the implementation is still immature. In time, Java should improve significantly, the programming community will become familiar with these "new" features and they will demand them of whatever succeeds Java. First, we have to get them working in Java.

Apple has made a lot of noise about Java being important to their future success, and to ours. This makes sense. If Apple's hardware business dries up, and their OS business dries up, what's left for them to do with their valuable brand name? Apple has been positioning themselves to be an important player in the Java sandbox.

Even so, Apple's Java strategy is way behind that of the rest of the world. Currently, Mac developers have to wait far too long if they want to use the latest Java technologies on the Mac. Many developers don't wait, they use another OS. Hopefully, Apple is fixing this problem; only time will tell.

MacHASP Packs More Into Less.

Based on state-of-the-art ASIC technology, MacHASP packs the industry's most advanced security, compatibility and flexibility into a compact and end-user-friendly dongle.

Grow With Aladdin!

The fastest growing company in the industry, with over 4 million keys sold to 20 thousand developers worldwide, Aladdin is setting the standard for software security today.

NSTL Study Rates HASP No.1!

A recent test conducted by the National Software Testing Labs[†], the world's foremost independent lab, compared the flagship PC products of leading software protection vendors. The result? HASP was rated the clear overall winner - and number one in all the major comparison categories. For a full copy of the NSTL report, contact your local HASP distributor.

MacHASP® PROTECTS MORE.

Mac™OS

These days, more and more developers are choosing to protect their software against piracy. They're protecting more products, on more platforms, with better protection – and selling more as a result.

And more of these developers are protecting with MacHASP. Why? Because MacHASP offers more security, more reliability and more features than any other product on the market. Only MacHASP offers capabilities such as network support, anti-debugging envelope protection, and secure remote activation and updating.

MacHASP supports the most advanced platforms, including all versions of MacOS and Power Mac – as well as AppleTalk networks. And because Aladdin is a licensed Apple Partner, MacHASP guarantees full, transparent compatibility with the ADB standard.

To learn more about how you can protect better – and sell more – call now to order your MacHASP Developer's Kit.

The MacHASP Developer's Kit contains everything you need to protect your software today!

1-800-223-4277

www.aks.com

ALADDIN™

The Professional's Choice

North America	**Aladdin Knowledge Systems Inc.** Tel: (800) 223 4277, 212-564 5678, Fax: 212-564 3377, E-mail: hasp.sales@us.aks.com
Int'l Office	**Aladdin Knowledge Systems Ltd.** Tel: +972-3-636 2222, Fax: +972-3-537 5796, E-mail: hasp.sales@aks.com
Germany	**FAST Software Security AG** Tel: +49 89 89 42 21-37, Fax: +49 89 89 42 21-40, E-mail: info@fast-ag.de
United Kingdom	**Aladdin Knowledge Systems UK Ltd.** Tel: +44 1753-622266, Fax: +44 1753 622262, E-mail: sales@aldn.co.uk
Japan	**Aladdin Japan Co., Ltd.** Tel: +81 426-60 7191, Fax: +81 426-60 7194, E-mail: sales@aladdin.co.jp
Benelux	**Aladdin Software Security Benelux B.V.** Tel: +31 24-641 9777, Fax: +31 24-645 1981, E-mail: 100526.1356@compuserve.com

■ **Aladdin Russia** 095 9230588 ■ **Australia** Conlab 03 98985685 ■ **Chile** Micrologica 02 7350041 ■ **China** Shanghai URI 021 64377828 ■ **Czech** Atlas 02 766085 ■ **Denmark** Berendsen 039 577316 ■ **Egypt** Zeineldein 02 3604632 ■ **Finland** ID Systems 0 8703520 ■ **France** 1 40859885 ■ **Greece** Unibrain 01 6756320 ■ **Hong Kong** Hastings 02 5484629 ■ **India** Solution 011 2148254 ■ **Italy** Partner Data 02 26147380 ■ **Korea** Dae A 02 8484481 ■ **Mexico** SiSoft 91800 55283 ■ **New Zealand** Training 04 5666014 ■ **Poland** Systherm 061 480273 ■ **Portugal** Futurmatica 01 4116269 ■ **Romania** Ro Interactive 064 140283 ■ **Singapore** ITR 065 5666788 ■ **South Africa** D Le Roux 011 8864704 ■ **Spain** PC Hardware 03 4493193 ■ **Switzerland** Opag 061 7169222 ■ **Taiwan** Teco 02 5559676 ■ **Turkey** Mikrobeta 0312 4670635 ■ **Yugoslavia** Asys 021 623920

by Dave Mark, ©1997, All Rights Reserved

A Cool Rhapsody Web Site & Navigating the OPENSTEP Doc

Hopefully, by the time you read this column, you've got a brand-spanking-new version of Rhapsody humming along on your Power Macintosh. My fingers are crossed! Either way, be sure to check out John Norstad's excellent Rhapsody web site at <http://charlotte.acns.nwu.edu/jln/wwdc97.html>.

The name of the site is the "1997 Apple Worldwide Developers Conference (WWDC) NUMUG Trip Report, A Rhapsodic Adventure". Don't let the name fool you. This site is much more than a WWDC rehash. First, it does an excellent job of reviewing the conference from a technical perspective, but it goes further than that. It presents a truly understandable picture of Rhapsody from soup to nuts.

The site starts off with a brief introduction, then launches into a discusssion of Apple's dual OS strategy (Mac OS lives!) and delivery schedules. This is followed by John's "good news, bad news, good news" perspective on Apple's problems. Next is a detailed Rhapsody Architecture Overview, complete with architecture diagrams to help you keep things straight and a discussion of Apple's cross platform strategy.

John graciously allowed me to include some of the material from his site in this column. Obviously, by the time you read this, the story will have changed somewhat. I encourage you to visit John's site and get the latest and greatest. And, if you see John at a conference (he goes to all of them), be sure to thank him for a job well done.

John Norstad on the Blue Box

As you know by now, Apple's new Macintosh architecture includes a Blue Box, which represents the classic Mac OS rehosted on the Rhapsody core OS, and a Yellow Box, which represents the new, OPENSTEP based technology, also hosted on the Rhapsody core OS. Here's some of what John had to say about the Blue Box:

"Blue provides much better compatibility than Copland. For example, almost all extensions work fine in Blue. Copland would have broken all extensions.

Playing with Blue in the lab was actually quite boring, since it was almost indistinguishable from a plain Mac running Tempo. That's the whole point, but it was still boring!

In terms of compatibility for old Mac OS software, the transition from Mac OS to Rhapsody should be very similar to the very successful transition from 68K to PowerPC several years ago. Most software will "just work". There will be a few problems and exceptions, but they should be minor.

The only old software that will break under Blue are programs which talk directly to hardware without going through the Device Manager, some kinds of extensions which patch File Manager traps and expect to be able to intercept all file system I/O (Yellow Box file I/O to shared disk volumes will not be intercepted by these kinds of Blue Box patches), and any other software that modifies or relies on the internals of shared system services.

Blue is not an emulator of any kind. It is mostly an exact copy of today's Mac OS, bug-for-bug, feature-for-feature, just rehosted on the new core OS. Most programs should run just as fast as they do on Mac OS, or perhaps only a little bit slower. Some operations will even run faster, due to performance improvements in the core OS.

Blue is very similar to MAE (the "Macintosh Application Environment"), an Apple product which lets you run Mac software on UNIX systems. Blue is simpler than MAE because it does not require a PowerPC emulator. Blue uses a RAM-based ROM image. There's no hardware ROM.

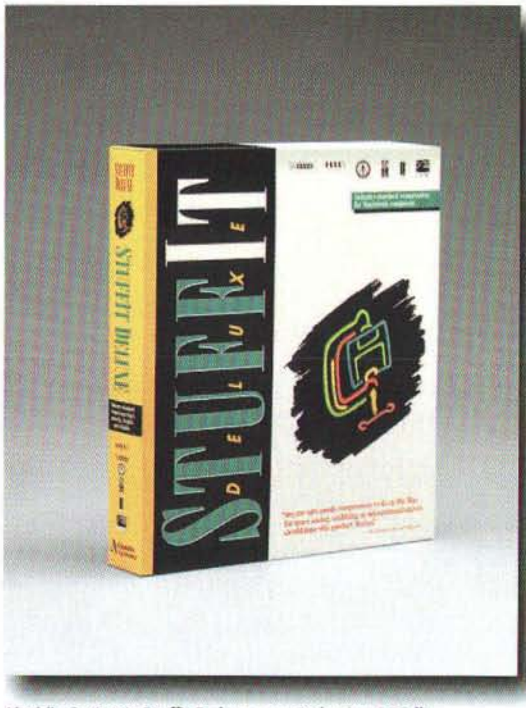
There is no preemptive multitasking or protected memory inside the Blue Box. A Blue application that crashes can still take down the entire Blue Box, just like an errant application on Mac OS can take down the entire Mac. An errant Blue application, however, cannot crash the core OS or Yellow Box. In Rhapsody, if a Blue crash occurs, you can easily and quickly reboot just the Blue Box, without having to restart the entire computer.

Behind every good Mac app...

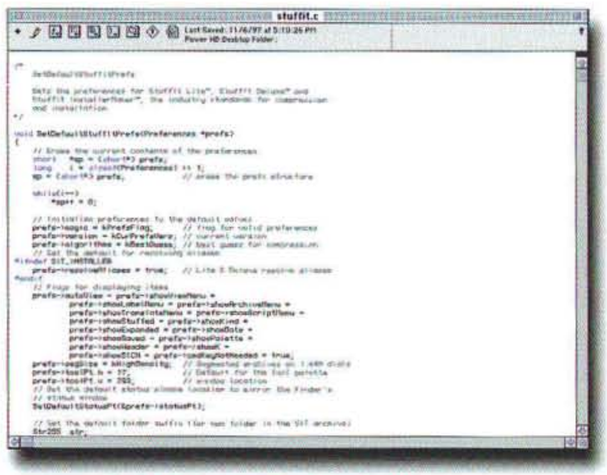
Every job requires the proper tool. When it comes to crafting good code, **BBEdit 4.0** is the **Swiss Army Chainsaw** of text editors.

"If you have to spend time with text on the Mac, you should use BBEdit."
-- MacWeek, September 30, 1996

- **Compare** source files and **apply differences**
- **Open From** and **Save To FTP**
- **Search** and **replace** multiple files using **Grep**
- Navigate source files with integrated **PopupFuncs™** technology
- **Browse projects**



Aladdin System's **StuffIt Deluxe**, created using **BBEdit**.



```
/*
    SetDefaultStuffItPrefs

    Sets the preferences for StuffIt Lite™, StuffIt Deluxe™ and
    StuffIt InstallerMaker™, the industry standards for compression
    and installation
*/

void SetDefaultStuffItPrefs(Preferences *prefs)
{
    // Erase the current contents of the preferences
    short   *sp = (short*) prefs;
    long    i = sizeof(Preferences) >> 1;
    sp = (short*) prefs;            // erase the prefs structure

    while(i--)
        *sp++ = 0;

    // Initialize preferences to the default values
    prefs->magic = kPrefsFlag;       // flag for valid preferences
    prefs->version = kCurPrefsVers;  // current version
    prefs->algorithms = kBestGuess;  // best guess for compression
    // Set the default for resolving aliases
#ifndef SIT_INSTALLER
    prefs->resolveAliases = true;    // Lite & Deluxe resolve aliases
#endif
    // Flags for displaying items
    prefs->autoView = prefs->showViewMenu =
            prefs->showLabelMenu = prefs->showArchiveMenu =
            prefs->showTranslateMenu = prefs->showScriptMenu =
            prefs->showStuffed = prefs->showKind =
            prefs->showExpanded = prefs->showDate =
            prefs->showSaved = prefs->showPalette =
            prefs->showHeader = prefs->showC =
            prefs->showSICN = prefs->cmdKeyNotNeeded = true;
    prefs->segSize = kHighDensity;  // Segmented archives on 1.4MB disks
    prefs->toolPt.h = 37;           // Default for the tool palette
    prefs->toolPt.v = 283;          // window location
    // Set the default status window location to mirror the Finder's
    // status window
    SetDefaultStatusPt(&prefs->statusPt);

    // Set the default folder suffix (for new folder in the SIT archive)
    Str255  str;
```

Source code from **StuffIt Deluxe** displayed in **BBEdit**.

...is a great text editor.

BBEdit 4.0

MacUser AWARDS Editors' Choice 1996 Winner · MacUser October 1996 · score card MacWEEK ◆◆◆◆◆

http://www.barebones.com/chainsaw
To order, call 617-778-3100

Bare Bones Software, Inc.
P.O. Box 1048, Bedford, MA 01730 • main 617-778-3100 • fax 617-778-3111

To the Mac OS running inside Blue, it appears as if virtual memory is turned off on a Mac with 1 gigabyte of memory! The core OS does virtual memory operations behind the scenes, but this is mostly transparent to the Blue Box. There are two important benefits of this new scheme:

- No more worries about memory fragmentation!
- You can set memory partition sizes very large with no ill-effects for most applications.

The Blue Box will also provide improved stability via "guard pages". These are special virtual memory pages that are marked read-only and are placed at the beginning and end of each program's stack and heap. In the regular Mac OS, programs sometimes "run off the end" of their stack or heap when writing memory, and end up trashing memory belonging to the system or other programs. This often causes bad crashes. In Blue, such a misbehaving program will crash, but it won't take down other programs or the whole Blue Box along with it.

Several people asked about the following obvious idea: How about having multiple Blue Boxes? This would give some of the benefits of preemptive multitasking and protected memory to Mac OS programs. Apple's answer is that there are problems that would have to be solved. For example, each copy of the Blue Box expects to have exclusive write access to desktop database files. Also, the memory footprint would be rather outrageous. While it might be possible to do this in the future, it is unlikely that this feature will be included in Rhapsody's Unified Release.

In summary, the Blue Box provides excellent compatibility for Mac OS software, comparable performance to the regular Mac OS, even better performance in some cases, and somewhat better stability.

John Norstad on the Yellow Box

Here's a snippet from John's discussion of the Yellow Box.

In addition to its OPENSTEP foundation, the Yellow Box includes the following advanced technologies from NeXT.

- Multi-user system with security. Rhapsody can be used as a single-user system much like today's Mac OS, or you can have multiple UNIX usernames and passwords with UNIX file security. Yes Virginia, you'll be able to telnet to your Rhapsody box!
- PDO = Portable Distributed Objects. A state-of-the-art system for distributed computing. Objects running in separate processes on separate computers can send each other messages almost as effortlessly as they do when they are in the same program. One of the NeXT engineers who designed and implemented PDO presented at the conference sessions. I'm quite impressed by PDO.
- EOF = Enterprise Object Framework. EOF provides an object interface to all of the major commercial database products. It's important in the enterprise market.
- WebObjects. This is a system for rapid development of web pages, CGI gateways, and web server plug-ins.

The Yellow Box will also incorporate several major technologies from Apple:

- The QuickTime media layer (QTML): QuickTime, QuickDraw 3D, and QuickTime VR.
- QuickDraw GX typography.
- ColorSync.
- Scripting. At WWDC, Apple hadn't decided exactly what the scripting language would look like (AppleScript or otherwise), but they promised Apple events and a ubiquitous scripting system integrated into the OPENSTEP classes. After WWDC, the trade press reported that Apple decided to support AppleScript in the Yellow Box.
- V-TWIN search engine.

Applications running in the Yellow Box enjoy the full benefits of all the services provided by the Mach kernel, including preemptive multitasking and protected memory. We'll talk about this more later when we go into more detail about the core OS.

Go Visit the Site!

The previous two sections represent the tip of the iceberg as far as this site goes. There are sections on Yellow and Blue integration, Rhapsody advanced look and feel, technical issues to be resolved and, of course, the site will be updated by the time this issue hits the street. Spend the time to go through the entire site. Again, that URL is <http://charlotte.acns.nwu.edu/jln/wwdc97.html>.

The OPENSTEP Librarian

One of the most important features of OPENSTEP that will likely make its way into Rhapsody is OPENSTEP's documentation system, known as the Librarian (old time NeXT programmers know the Librarian as the "Digital Librarian" — Librarian is the current term).

On the dock, look for an icon that looks like a series of six books, one next to the other (**Figure 1**). If you don't have the Librarian in your dock, you'll find the application in the directory /NextApps/Librarian.app.

Figure 1. *The Librarian icon. The Librarian application is found in /NextApps/Librarian.app.*

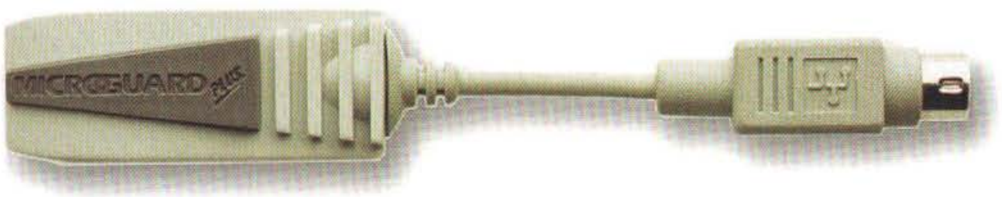
- Launch the Librarian application.

The window that appears (**Figure 2**) is known as a bookshelf. A bookshelf is a collection of indexable, searchable documents. A series of examples will make this clear.

You already know that...

MicroGuard Copy Protection is -

UNBEATABLE



SEE YOU AT:

COMDEX Fall '97
November 17-21, 1997
LAS VEGAS

Booth #S8049

So... Here's how you can reach us:

www.micromacro.com

International
Micro Macro Technologies, Ltd.
3 Hashikma St.
P.O.Box 11516, Azur 58001
Israel

Tel: (972-3) 558-2345
Fax: (972-3) 558-2344
E-mail: info@micromacro.com

USA
MicroGuard
631 South Pontiac St.
Denver CO. 80224
USA

Tel: (303) 320-1628
Fax: (303) 320-1599
E-mail: usa@micromacro.com

Mac™OS

MICROSOFT® WINDOWS™ COMPATIBLE

MICROGUARD PLUS™

Charter member
Pirate Busters

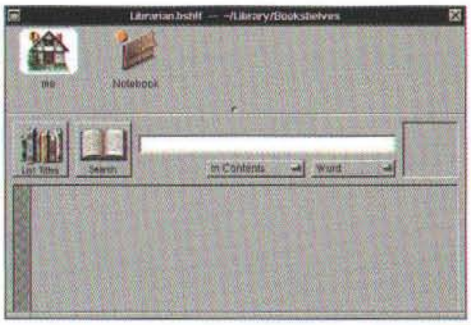


***Figure 2.** The bookshelf window that appears when the Librarian is opened.*

- In the Librarian application, select Open... from the Bookshelf menu.
- Navigate all the way up and into the directory /NextLibrary/Bookshelves.

The /NextLibrary directory is the default place for storing system resources such as application preferences. Since UNIX is a multi-user OS, there's also a way to store preference-type files on a per user basis, in the user's home directory in a subdirectory called Library. For example, the global bookshelves are in /NextLibrary/Bookshelves/, your own bookshelf is in /me/Library/Bookshelves/.

By default, you'll find 4 bookshelf files in the /NextLibrary/Bookshelves directory (see **Figure 3**): DevTools.bshlf, EntrepriseObjects.bshlf, NextDeveloper.bshlf, and SysAdmin.bshlf.

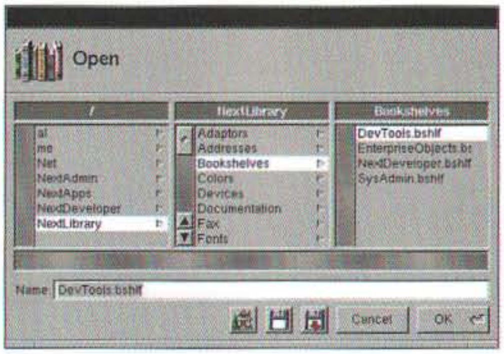


***Figure 3.** The four bookshelves in the /NextLibrary/Bookshelves/ directory.*

- In Librarian's Open dialog, select DevTools.bshlf and click the Open button.

The DevTools bookshelf window will appear (**Figure 4**). In the top half of the window, you'll see 7 icons. Each icon represents a directory somewhere on your hard drive. The documents in that directory combine to make up a book. The Librarian can be used to search a book or set of books, based on a variety of criteria. If a book's directory contains any sub-directories, the docs and directories in those sub-directories are part of the tree that makes up the book.

To rephrase this, a bookshelf is made up of a series of books. Each book represents a tree of documents contained within a specified directory.

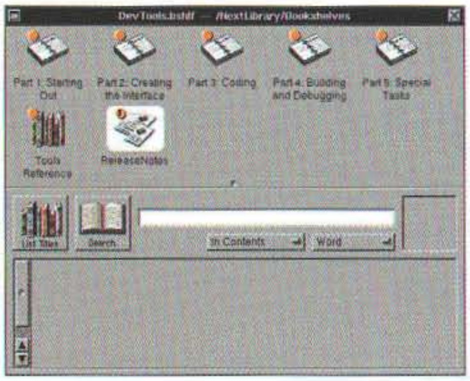


***Figure 4.** The DevTools bookshelf window.*

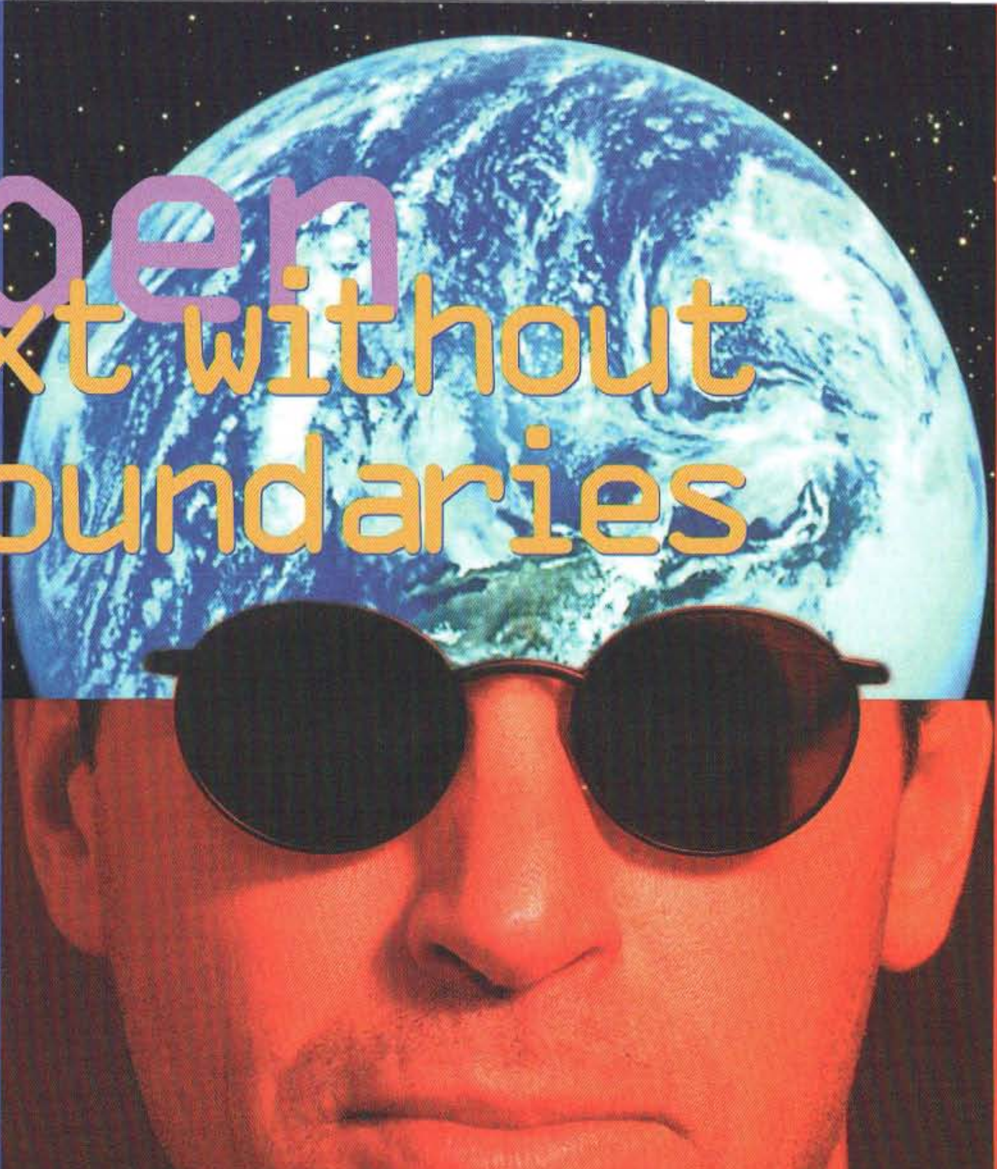
Librarian allows you to build an index for a book. The index makes searching the book significantly faster than a search of the book without an index. You can tell if a book has an index by looking at the book's icon in the bookshelf window. If the icon has a yellow dot with an upper-case "I" in the upper-left corner, the book is indexed. A solid yellow dot indicates that while the book itself is not indexed, it is part of a book that *IS* indexed. In other words, the book's directory is somewhere in the tree of an indexed book. Books within books. Cool!

If a book's icon has no yellow dot at all, the book is not pre-indexed, and searches are done in a much slower, grep-like fashion. Basically, if you plan on searching a book on a regular basis, make sure the book (or its ancestry) is indexed.

- In the DevTools bookshelf window, double-click the Tools Reference icon.

The Target Inspector window appears (**Figure 5**). The Target Inspector allows you to modify the attributes of the selected book. Notice that the name of the book, Tools Reference, appears in bold next to the book's icon towards the top of the Inspector window. Underneath the icon and book name is a path, showing you how to get to the book's directory. Note that this is a partial path name. In this case, the full path was /NextLibrary/Documentation/NextDev/Reference/DevTools. Note that the name of the book isn't necessarily the same as the name of its directory.

Visit MacTech® Magazine's Web site!
http://www.mactech.com



wide

sesame

door

mind

season

architecture

for business

possibilities

horizons

house

now

platform

window

road

sky

borders

up

CONSIDER the possibilities. They're wide open. Simply put, OpenPaige™ can do anything you want to the text of your software application.

Across platforms, across architectures, even UNIX™. OpenPaige is a feature-rich text environment. Every aspect of text and layout formatting, editing and display is possible. Stylized text. Scaling. Container and non-rectangular shapes. Style sheet support. Cut, copy and paste. Embedded pictures, objects, buttons and even calls to other applications are at your fingertips.

Don't waste time struggling with your text. We've already done the work for you. Drop in OpenPaige and go. Use as much or as little of it as you wish. Available as object code or as source code, it has an elegantly small footprint. Yet it will empower you to open doors, boundaries, and minds with your software application.

But don't take our word for it. Check out our extensive and diverse client list. They've all discovered how the robust and flexible power of OpenPaige can open up entire new worlds of possibilities.

Open Paige™

WITHOUT US THEY WOULDN'T EXIST (OR AT LEAST THEY'D BE MUCH MORE FRUSTRATED.)

America Online

Apple Computer

Broderbund Software

Canon

Claris

CompuServe

Dataware

Digital Equipment

Fujitsu

GE Information Systems

ichat

Macromedia

National Geographic

Now Software

QualComm

Seagate Software

3M

US West

Xerox

Datapak

For additional information, check out our web site at www.datapak.com/~datapak/ or call us at 1-800-327-6703.

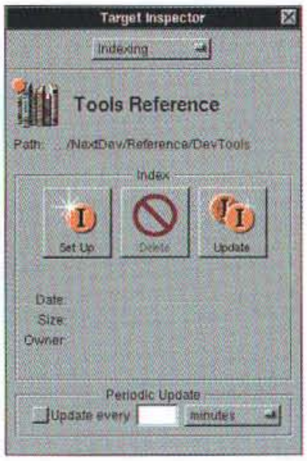


Figure 5. *The Target Inspector window, showing the index information.*

- Select Indexing from the Target Inspector's popup menu.

An Index pane will appear in the center of the window and a Periodic Update pane will appear in the bottom of the window. There are three buttons in the index panel: Set Up, Delete, and Update. Use these to create a new index, to delete an existing index, and to update an index, making sure it reflects the current content's of the book's directory. Note that updating an index is much faster than deleting and recreating an index.

You can't create an index for a directory that is part of another index. Then again, why would you since an index already exists!

The Periodic Update checkbox lets you specify that the system should update your index automatically, every X minutes or X hours. This is really nice if you've indexed a project's source code (good idea!) and want to update your source code index every night, for example!

- Go back to the DevTools.bshlf window.
- Double-click the ReleaseNotes icon.

This will bring up the inspector window. Note that the Set Up button is disabled (this book already has an index) and that the Delete button is enabled. DO NOT delete this index. It is a useful one to have. If you do have the time and the inclination, go ahead and delete the index and then click the Set Up button to recreate it. This will give you a sense of how long this can take.

Note that you can shift-click to select multiple icons in a bookshelf window. Index commands apply to all the selected books.

- Back in the DevTools bookshelf window, shift-click to select all 7 book icons.
- Click the List Titles button.

Underneath the text field, Librarian will tell you how many titles it found. In my case, the Librarian found 199 documents inside all 7 directories.

- Type the word "action" in the text field, then click the Search button or hit enter/return.

You'll find around 38 documents, each of which contains the word action. Very fast search, eh? The two popups underneath the text field allow you to constrain your search. You can search "In Contents" or "In File Name". The second popup lets you search using Word, Prefix, Within, Literal, and Expression. For the moment, Word is fast, the others are a bit slower.

- Close the bookshelf window.

CREATING A NEW BOOKSHELF

- In Librarian, select New from the Bookshelf menu.

A new empty bookshelf window will appear.

- Go to the Workspace application, into a File Viewer window, and navigate to the directory /NextLibrary/Frameworks.

The AppKit and Foundation frameworks have associated documentation which is already indexed. We're going to add these indexes to this new bookshelf.

- In your File Viewer, continue descending into AppKit framework/Resources/English.lproj/Documentation/Concepts.
- Back in Librarian, drag the new bookshelf window so you can see it and the File Viewer at the same time.
- In the File Viewer, drag the Concepts icon from the shelf into the top portion of the new bookshelf window.
- Repeat this for the Reference and ReleaseNotes directories.

Cogratulations — you've just created a bookshelf containing three books. Note that all three directories are already indexed.

Why create a bookshelf? Remember, each book is basically an indexable, searchable representation of all the documents in a specific directory. You can build a book based on a directory filled with 100 source code files. Want to search the source code in that directory? Use the Librarian's Search button to search the book's index at blazingly fast speeds. As you make your way through the rest of the column, you'll see what I mean. But there's no substitute for actually trying this out for yourself.

- In Librarian, select Save from the Bookshelf menu.

Though you can save the bookshelf wherever you like, stick to the standard and save it in your home directory in the Library/Bookshelves subdirectory. I saved mine in the directory /me/Library/Bookshelves/. When you enter a name for the new bookshelf, if you leave off the .bshlf, Librarian will tack it on for you.

In the UNIX universe (and NEXTSTEP is based on UNIX), a single machine can support multiple users, each of whom has their own account. Each user also has their own home directory. On my machine, my home directory is /me. If your home directory was /usr/puckett, you'd likely store your bookshelves in /usr/puckett/Library/Bookshelves.

Suppose we wanted to search the AppKit bookshelf we just created to learn how to create a new application.

- In the AppKit bookshelf window, select all three icons.
- Type the word application in the text field.
- Click the Search button. (I found around 200 matches.)
- Select In File Names from the left popup and do the Search again.

By selecting In File Names (as opposed to In Content), we've limited the search to documents with the word application in the title. Sometimes In Content will prove to be a better search and other times In File Names will work better.

In this case, searching In File Names produced only 2 matches. Much better! Don't worry if your numbers are different than mine. After all, we are likely running different versions of OPENSTEP on different machines with different configurations. The point is, if your search returns too many hits, try contraining the search in some way.

- We've just created and searched an AppKit bookshelf. Repeat the above procedure to create a FoundationKit bookshelf in the same directory.

In broad terms, the AppKit is all the interface related classes and the FoundationKit is non-interface related classes, such as File handling, Exception handling, NS Object, etc. To really see the difference, build the FoundationKit bookshelf and do a List Titles. Double-click on any title you like and it will open in Librarian.

- To add man pages to a bookshelf, go to myhost/NextLibrary/Documentation/ManPages/.
- Drag icon off shelf into bookshelf window. Voila. Searchable man pages.

Note the nr icon to the left of the document name. That means the doc is an nroff file. If the icon has a c it is a .c file. There are icons for rtf, folders, and lots of others. Just as it would on your Mac, when you double-click a title or icon, the appropriate application will be launched (or brought to the front) and the document will be opened. If you double-click a directory, the Workspace will pop to the front and a new File Viewer will open for that directory.

Indexing Your Source Code

- Back in Librarian, create a new bookshelf.
- In a File VIewer, find /NextDeveloper/Examples.
- Drag the Examples icon to the bookshelf window.
- In the File Viewer, find /NextDeveloper/Examples/AppKit/TextEdit.
- Drag its icon to the bookshelf. Drag any others you like.

Note that TextEdit has solid yellow circle while Examples has a Shift-I circle. Examples is the directory with the index and the ancestor directory to TextEdit. TextEdit searches use the Examples index. Even so, TextEdit searches will only find files in the TextEdit area. The search won't be quite as fast as if it had its own index, but the search will still be pretty fast!

- Save your new bookshelf.

So now we've got a bookshelf with a bunch of interesting sample source code. Suppose you wanted to find out how to save a document. You know that this source code does that.

- Select the TextEdit icon, type SaveDocument (one word) in the text field, then click Search.

The search returns three documents. The routine we want is in Document.m.

- Double-click AppKit/TextEdit/Document.m.

Librarian will open the file for you. You can click the Find button several times to find the SaveDocument routine. If you scroll down a bit to an else clause, you'll find the term textStorage. To search on this term, select it, copy it, then paste it into your bookshelf text area and click the Search button. This is not as nice as ProjectBuilder or CodeWarrior's find facility, but it is a very nice way to explore, to get your head around a new concept.

The Easy way to Create a new Bookshelf

- In a File Viewer window, navigate into /NextDeveloper/Examples/AppKit and select the AppKit icon.
- Select Target from the Services/Librarian submenu.

Librarian comes forward and creates a new bookshelf window containing the AppKit directory. That's it! You are now ready to search.

- Launch the Edit application.
- Create a new window, type the word textstorage, and select it.
- Select Search from the Services/Librarian submenu.

Librarian comes forward, and the frontmost bookshelf is searched for textstorage using the selected indexes. How cool is that? Notice that the Librarian submenu also features Update Index, which lets you update the index of the directory containing the file you are editing (which is really nice if you are editing source code!)

Till Next Month...

This is the easy way to use bookshelves. More importantly, you've just had your first taste of OPENSTEP's services facility. In this case, the Librarian offered a series of services such as target and search, and Update Index. We'll get into Services in detail in a future column. But for now, why not use the Librarian to do a little research on Services yourself! MT

by David Rand

Contour Plotting in Java

Developing a simple, useful applet using Metrowerks CodeWarrior Java

Introduction

This article discusses a Java applet which plots contours based on a given matrix of floating-point values. The idea is to present an algorithm which is simple enough to be implemented in a few hundred lines of code yet which performs a useful, non-trivial operation. Java is of course multi-platform, but this article focuses on development of the applet on the Macintosh, using Metrowerks CodeWarrior Java (specifically, CW11, which includes version 1.0.2 of the JDK). It is assumed that the reader is at least somewhat familiar with the Java language and the CodeWarrior environment. See, for example, Dave Mark's series of "Getting Started" articles [1] in MacTech magazine.

A contour plot is a convenient way of representing three-dimensional data on a two-dimensional surface such as a map or a computer screen. If you have ever gone hiking and took with you a topographical map indicating the lay of the land, you have used a contour plot. The lines of constant elevation are the contours. If the lines are close together then we have a region of steep slope, whereas widely spaced lines indicate more gradual slope. A closed contour looping back on itself indicates the existence of an extremum — a peak or a depression — somewhere inside the loop. If the extremum is pronounced, i.e. a sharp peak or a deep depression, then several such loops will be nested together concentrically.

The Plotting Algorithm

The algorithm used for contour plotting is taken from a 1978 article by W. V. Snyder [2]. The Java implementation was created by first translating Snyder's Fortran source code manually into C, then reworking the C code to remove the many convoluted `goto` constructs, and finally manually translating this "unravelled" C code into Java. The second step, necessary because Java does not support `goto` statements, proved to be the most laborious. This incompatibility should not be considered a limitation of Java; on the contrary, it is a result of the lack of appropriate block structures in old versions of Fortran.

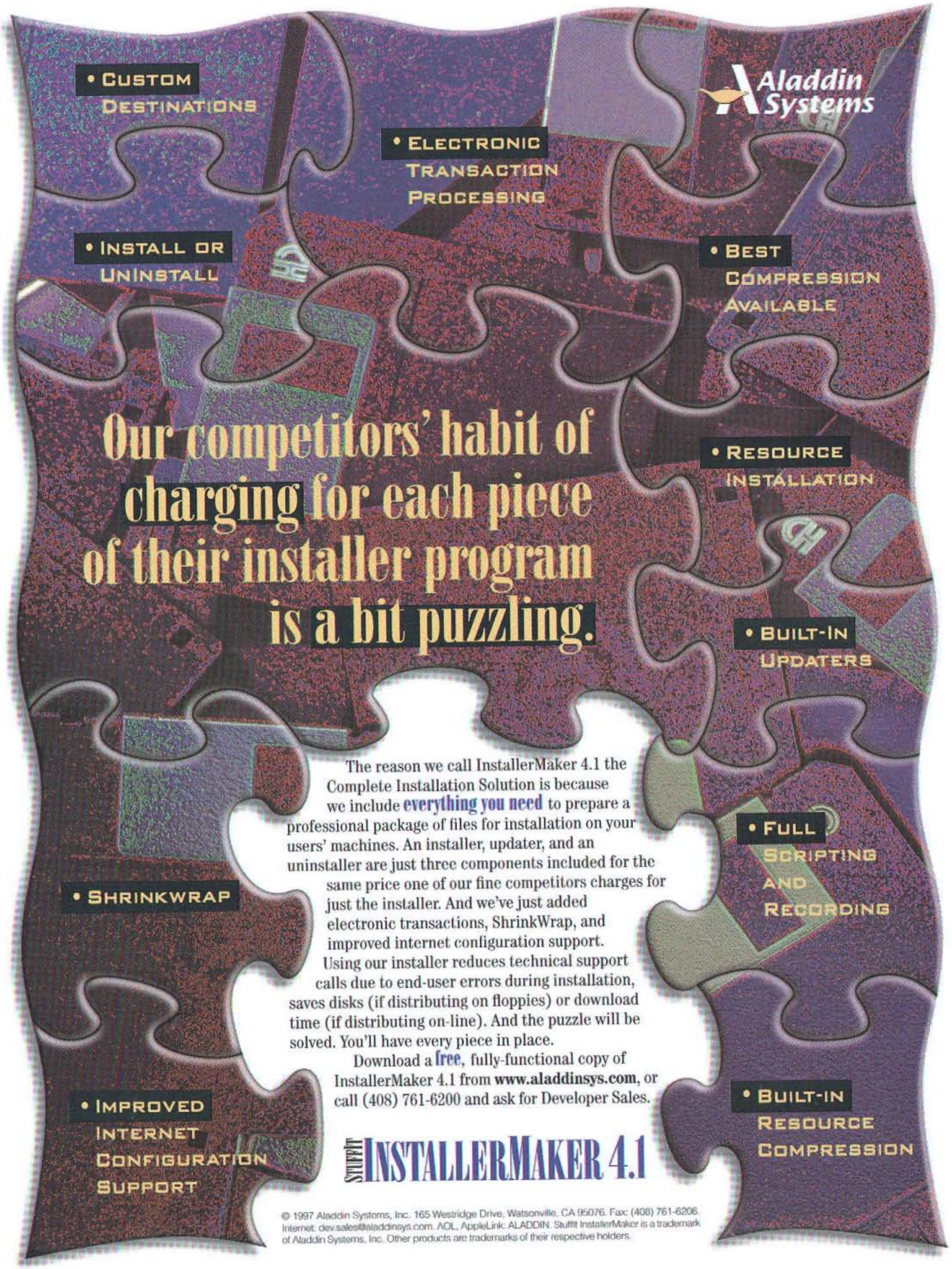
The plotting algorithm can be summarized thus:

a) Given a matrix of floating point values which are the values of a function `z = f(x,y)` given at the nodes of a grid of `x` and `y` values (the grid values are assumed equally spaced, although the horizontal and vertical spacing may differ), the program determines the minimum and maximum values of `z` and then computes a number of contour values (in this implementation, 10 values) by linear or logarithmic interpolation between the extrema.

b) The program "walks" about the grid of points looking for any segment (i.e. a line joining two adjacent nodes in the grid) which must be crossed by one of the contours because some contour value lies between the values of `z` at the nodes.

c) Having found such a segment, it finds the intersection point of the contour and the segment by linear interpolation between the nodes. It also stores the information that the current contour value has been located on the current segment, so that this operation will not be repeated.

David Rand works at the Centre de recherches mathématiques (CRM) at the Université de Montréal where he manages the CRM's web site. He has developed a number of Macintosh applications such as the concordance-editor *Concorder 3* and the shareware text- and list-editor *Zephyr 1.1*.


Aladdin Systems
• Custom Destinations
• Electronic Transaction Processing
• Install or Uninstall
• Best Compression Available
• Resource Installation
• Built-In Updaters
• Full Scripting and Recording
• Shrinkwrap
• Improved Internet Configuration Support
• Built-in Resource Compression

Our competitors' habit of charging for each piece of their installer program is a bit puzzling.

The reason we call InstallerMaker 4.1 the Complete Installation Solution is because we include **everything you need** to prepare a professional package of files for installation on your users' machines. An installer, updater, and an uninstaller are just three components included for the same price one of our fine competitors charges for just the installer. And we've just added electronic transactions, ShrinkWrap, and improved internet configuration support. Using our installer reduces technical support calls due to end-user errors during installation, saves disks (if distributing on floppies) or download time (if distributing on-line). And the puzzle will be solved. You'll have every piece in place.

Download a **free**, fully-functional copy of InstallerMaker 4.1 from **www.aladdinsys.com**, or call (408) 761-6200 and ask for Developer Sales.

STUFFIT INSTALLERMAKER 4.1

PLATFORMS: APPLE SYSTEM 7 • APPLE A/UX •

NT (ALPHA) • MIPS ABI (SGI) • AT&T UNIX • BANYAN • INTERACTIVE UNIX • LINUX • MOTOROLA 88OPEN • QNX •

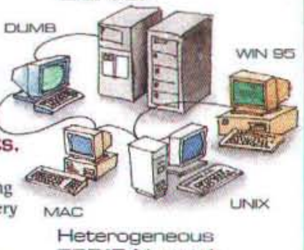
Q: What does it take t superior client/server
A: *A SUPERIOR S*

Java Interface! New c-tree 6.7! Enhanced Servers! Check it out!! www.faircom.com

START with the most advanced client-side SDK on the market: c-tree® Plus at $895.
- **Complete "C" Source code**
- ROYALTY FREE (Client Side)
- Multiple supported protocols
- Fast, portable, reliable
- Powerful features like transaction processing
- Win95, NT, and Windows 3.1 ready

RESU A solid, ecor easily deployable product that fits your needs.
- Portable
- Scalable
- Exceptional Performance
- Flexible
- Easy Server distribution
- Convenient OEM terms

FAIRCOM® Server



Heterogeneous TCP/IP Network

ADD a strong, multi-platform, industrial-strength Server that supports.
- File mirroring
- Heterogeneous networking
- Automatic disaster recovery
- Multi-threaded design
- Best price/performance available: from $445- $3745

You can't find a better client SDK with these features!
Over sixteen years of proven reliability and performance.
No one else supports over 30 platforms in this price range!

c-tree Plus®
- Complete C Source
- Single/Multi User
- Client/Server (optional)
- Full ISAM functionality
- No Royalties
- Transaction Processing
- Fixed/Variable Length Records
- High Speed Data/Index Caching
- Batch Operations
- File Mirroring
- Multiple Contexts
- Unsurpassed Portability

FairCom® Server
- Client/Server Model
- Transaction Processing
- Requires <2MB RAM
- Online Backup
- Disaster Recovery
- Rollback - Forward
- Anti-Deadlock Resolution
- Client-side "C" Source
- Multi-threading
- Heterogeneous networking
- File Mirroring
- OEM/Source Available

FOR YOUR NEXT PROJECT CALL FAIRCOM: YOU CAN'T FIND A BETTER HETEROGENEOUS CLIENT/SERVER SOLUTION!

Also inquire about these FairCom products:
d-tree™
r-tree®
ODBC Driver

FAIRCOM® CORPORATION

Since 1979

WWWeb Address: http://www.faircom.com/
800-234-8180

U.S.A. phone (573) 445-6833 fax (573) 445-9698
EUROPE phone (035) 773-464 fax (035) 773-806
JAPAN phone (0592) 29-7504 fax (0592) 24-9723

WIND S • NT • WINDOWS 95 • OS/2 • NLM • DOS • LYNX • SCO • SYSTEM MANAGER • SUN SPARC-SOLARIS • IBM RS/6000

• HP9000 • RS/6000 • SUN O/S 4.X •

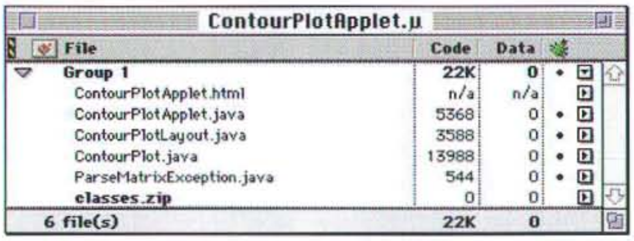
d) The program then attempts to locate a neighbouring segment having a similar property — that is, crossed by the same contour. If it finds one, it determines the intersection point as in step c) and then draws a straight line segment joining the previous intersection point with the current one. This step is repeated until no such neighbour can be found, taking care to exclude any segment which has already been dealt with.

e) Steps b), c) and d) are repeated until no segment can be found whose intersection with any contour value has not already been processed.

For a more detailed description of steps b) through e) of the algorithm, consult Snyder's article. Note that, as stated in step d), the path of each contour is constructed of linear segments, the simplest method possible. A more sophisticated algorithm, based on bicubic Hermite polynomials, may be found in [3].

THE CODEWARRIOR PROJECT

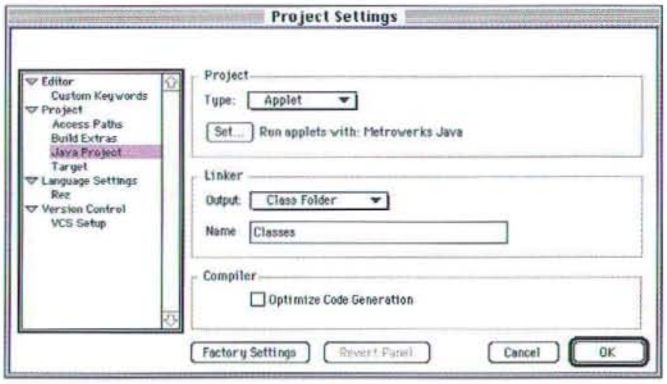
The Java applet discussed in this article was developed using Metrowerks CodeWarrior Java on a Power Macintosh. **Figure 1** shows the window corresponding to the project file named ContourPlotApplet.µ. **Figure 2** illustrates the project settings.



ContourPlotApplet.µ

File	Code	Data
Group 1	22K	0
ContourPlotApplet.html	n/a	n/a
ContourPlotApplet.java	5368	0
ContourPlotLayout.java	3588	0
ContourPlot.java	13988	0
ParseMatrixException.java	544	0
classes.zip	0	0
6 file(s)	22K	0

***Figure 1**. The CodeWarrior Project.*



Project Settings

Editor
  Custom Keywords
Project
  Access Paths
  Build Extras
  Java Project
  Target
Language Settings
  Rez
Version Control
  VCS Setup

Project
Type: Applet
Set... Run applets with: Metrowerks Java

Linker
Output: Class Folder
Name: Classes

Compiler
☐ Optimize Code Generation

***Figure 2**. The Project Settings.*

The project includes an html file, four java files (one for each new class) and the file classes.zip. The settings indicate that the project will generate an applet which will be run using the Metrowerks Java interpreter, and the classes (the applet as well as the three classes which it uses) will have their object code stored in class files in a folder called Classes inside the same folder as the project file. When the project is run, the interpreter will look in the html file for an <APPLET> tag from which it will read the dimensions and

MacTech® Now!

http://www.mactech.com

(...and we really mean now.)

Think of it as your source of news and announcements in the MacOS developer community. Loaded with up to the latest news, and constantly updated with developments in our industry.

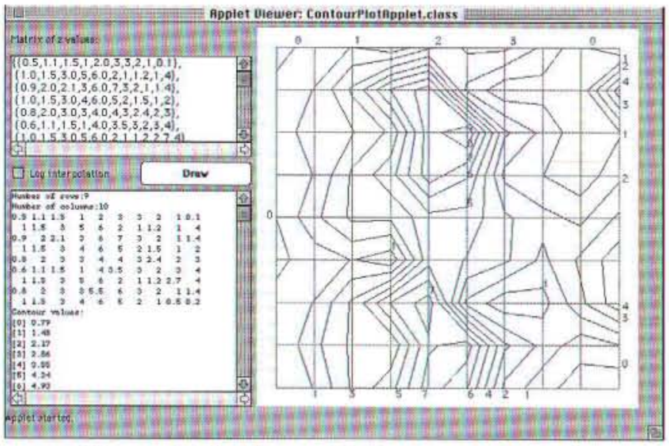
MacTech® NOW brings you up to speed on everything you need to know – instantly! And thanks to our new "fast-download" design, you'll get to the information you want in seconds.

Give it a spin! Check it out today and get access to over 1500 pages loaded with news, tips, programming secrets, product reviews, and much more. And for those of you looking for some kicks, there's the ever popular "Programmer's Challenge" section where you'll get to bang heads with the best in the community.

Log on to MacTech NOW. Things are happening right now, and you should be aware of them.

http://www.mactech.com

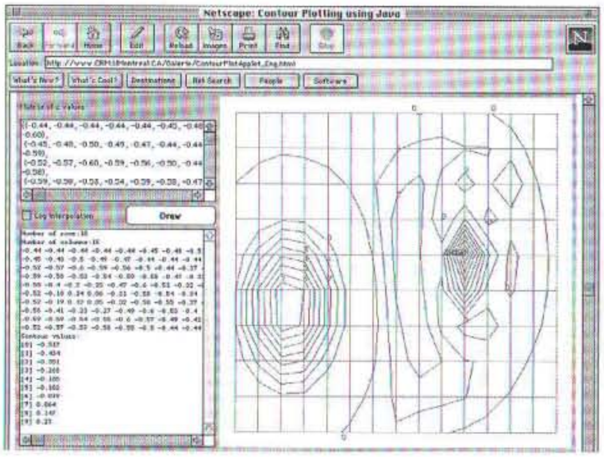
parameters of the applet. The html file can also be opened by a Java-enabled web browser to run the applet from within the browser; this will display the entire contents of the html file which may include instructions for using the applet or any related documentation.



```
{{0.5,1.1,1.5,1,2.0,3,3,2,1,0.1},
{1.0,1.5,3.0,5,6.0,2,1,1.2,1,4},
{0.9,2.0,2.1,3,6.0,7,3,2,1,1.4},
{1.0,1.5,3.0,4,6.0,5,2,1.5,1,2},
{0.8,2.0,3.0,3,4.0,4,3,2.4,2,3},
{0.6,1.1,1.5,1,4.0,3.5,3,2,3,4},
{1.0,1.5,3.0,5,6.0,2,1,1.2,2.7,4}
```

```
Number of rows:9
Number of columns:10
0.5 1.1 1.5  1   2   3   3   2    1 0.1
  1 1.5   3  5   6   2   1 1.2    1   4
0.9   2 2.1  3   6   7   3   2    1 1.4
  1 1.5   3  4   6   5   2 1.5    1   2
0.8   2   3  3   4   4   3 2.4    2   3
0.6 1.1 1.5  1   4 3.5   3   2    3   4
  1 1.5   3  5   6   2   1 1.2  2.7   4
0.8   2   3  3 5.5   6   3   2    1 1.4
  1 1.5   3  4   6   5   2   1  0.5 0.2
Contour values:
[0] 0.79
[1] 1.48
[2] 2.17
[3] 2.86
[4] 3.55
[5] 4.24
[6] 4.93
```

Figure 3. The ContourPlotApplet running in the Metrowerks Java interpreter.

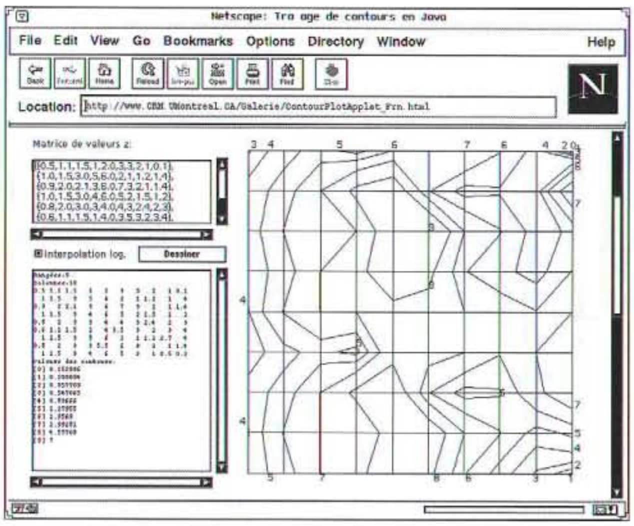


```
{{-0.44, -0.44, -0.44, -0.44, -0.44, -0.45, -0.48,
-0.60},
{-0.45, -0.48, -0.50, -0.49, -0.47, -0.44, -0.44,
-0.59},
{-0.52, -0.57, -0.60, -0.59, -0.56, -0.50, -0.44,
-0.58},
{-0.59, -0.59, -0.58, -0.54, -0.59, -0.58, -0.47
```

```
Number of rows:10
Number of columns:16
-0.44 -0.44 -0.44 -0.44 -0.44 -0.45 -0.48 -0.5
-0.45 -0.48 -0.5 -0.49 -0.47 -0.44 -0.44 -0.44
-0.52 -0.57 -0.6 -0.59 -0.56 -0.5 -0.44 -0.37
-0.59 -0.58 -0.53 -0.34 -0.50 -0.58 -0.47 -0.3
-0.55 -0.4 -0.2 -0.25 -0.47 -0.6 -0.51 -0.32
-0.52 -0.16 0.24 0.06 -0.21 -0.58 -0.54 -0.34
-0.52 -0.19 0.12 0.05 -0.32 -0.56 -0.55 -0.37
-0.56 -0.41 -0.23 -0.27 -0.49 -0.6 -0.53 -0.4
-0.59 -0.59 -0.54 -0.55 -0.6 -0.57 -0.49 -0.42
-0.52 -0.37 -0.33 -0.58 -0.55 -0.5 -0.44 -0.44
Contour values:
[0] -0.517
[1] -0.434
[2] -0.351
[3] -0.268
[4] -0.185
[5] -0.102
[6] -0.019
[7] 0.064
[8] 0.147
[9] 0.23
```

Figure 4. The ContourPlotApplet running in Netscape Navigator 3.0 Gold on a Macintosh.

Figure 3 shows the applet running in the Metrowerks Java interpreter. **Figure 4** shows the same applet (although using different data) running as part of a page at the web site of the CRM, Université de Montreal; the web server is a UNIX platform and the web client, as shown in the figure, is Netscape Navigator running on a Power Macintosh. To serve the applet in this way, all that is required in the UNIX file system is to create a directory called Classes in the same directory as that containing the html file and then to ftp (as raw data) the four class files from the Classes directory on the Macintosh where they were developed to the Classes directory on the UNIX machine.



```
{{0.5,1.1,1.5,1,2.0,3,3,2,1,0.1},
{1.0,1.5,3.0,5,6.0,2,1,1.2,1,4},
{0.9,2.0,2.1,3,6.0,7,3,2,1,1.4},
{1.0,1.5,3.0,4,6.0,5,2,1.5,1,2},
{0.8,2.0,3.0,3,4.0,4,3,2,4,2,3},
{0.6,1.1,1.5,1,4.0,3.5,3,2,3,4},
```

```
Rangées:9
Colonnes:10
0.5 1.1 1.5  1   2   3   3   2    1 0.1
  1 1.5   3  5   6   2   1 1.2    1   4
0.9   2 2.1  3   6   7   3   2    1 1.4
  1 1.5   3  4   6   5   2 1.5    1   2
0.8   2   3  3   4   4   3 2.4    2   3
0.6 1.1 1.5  1   4 3.5   3   2    3   4
  1 1.5   3  5   6   2   1 1.2  2.7   4
0.8   2   3  3 5.5   6   3   2    1 1.4
  1 1.5   3  4   6   5   2   1  0.5 0.2
Valeurs des contours:
[0] [illegible]
[1] [illegible]
[2] [illegible]
[3] [illegible]
[4] [illegible]
[5] [illegible]
[6] [illegible]
[7] [illegible]
[8] [illegible]
[9] 7
```

Figure 5. The ContourPlotApplet running in Netscape Navigator 3.01 on a UNIX platform.

As an illustration of platform independence, **Figure 5** also shows the applet running in Netscape Navigator on a UNIX platform. The data here are the same as in **Figure 3** except that logarithmic interpolation has been chosen. In addition, the eagle-eyed reader will notice that the character strings — such as the prompt in the upper left corner of the applet, the button, etc. — are in French. This is easily accomplished without modifying the applet, simply by changing the parameters in the <APPLET> tag in the html file. This requires, of course, that any language-dependent elements not be hard-coded in the Java source code. This approach is familiar to Macintosh programmers, who know that strings and such should be stored in an application's resource fork and not hard-coded in the application.

As shown in **Figures 3**, **4** and **5**, there are six user-interface elements in the applet, each implemented as a Java object of some type derived from the Component class. The contour plot is the largest component and is located in the right-hand part of the applet's rectangle. The other five components, all located in the left-hand part, are

1. The prompt "Matrix of z values:".
2. The data area in which the user enters the matrix of z values. Each row of the matrix is a list of floating-point values separated by commas and enclosed in brace brackets. Such as {0.5, 1.1, 1.5}. The rows also are separated by commas and the list of rows is enclosed in brace brackets. This format is similar to Mathematica syntax, for example. The rows need not all be of the same length; the program will complete any short rows with the appropriate number of zeroes.
3. The check box for choosing logarithmic instead of linear interpolation for the computation of the contour values. Logarithmic interpolation is possible only if all the z values are positive. In particular, logarithmic interpolation is unavailable if the rows of the matrix are not all of the same length, because short rows are filled with zeroes.

4. The "Draw" button. When the user clicks this button, the program parses the matrix in component #2, draws the contour plot if the data is valid, and finally shows some results (or an error message) in component #5.
5. The results area in which the applet displays some information about the plot just drawn. If the data are not valid for some reason, an error message will appear here. Otherwise, this area will display the number of rows and columns in the grid, the matrix of z values (with some rows extended by zeroes if necessary to make the matrix rectangular), and finally the ten contour values, numbered from 0 through 9.

The Source Code

The source code consists of an html file and four java files, one for each newly defined class. Each file has the same name as the class whose source code it contains, with the extension java appended. Thus the file ContourPlotApplet.java contains the class ContourPlotApplet, etc. Flanagan [4] is an invaluable reference for information about Java classes predefined in the Java API version 1.0, from which the classes discussed here are derived.

ContourPlotApplet is the main class of the four and is derived from java.applet.Applet. The class ContourPlotLayout is used to layout the user-interface items in the applet's rectangle. ContourPlot contains the code, adapted from Snyder [2], whose task is to draw the applet's most important component, the contour plot itself. Finally, ParseMatrixException is used to signal error conditions corresponding to invalid data.

ContourPlotApplet.html

This html file contains the <APPLET> tag which declares the CODEBASE (that is, the folder where class files are located), the CODE (that is, the name of the applet class), the graphical dimensions of the applet and its parameters. This <APPLET> tag is shown in Listing 1; for brevity, some of the parameters are omitted from the listing.

Listing 1

The <APPLET> tag in ContourPlotApplet.html

```
<APPLET CODEBASE="./Classes/"
    CODE="ContourPlotApplet.class" WIDTH=715 HEIGHT=460>
<PARAM NAME="stringX"           VALUE="Number of rows:">
<PARAM NAME="stringY"           VALUE="Number of columns:">
<PARAM NAME="stringZ"           VALUE="Matrix of z values:">
<PARAM NAME="stringBox"         VALUE="Log interpolation">
<PARAM NAME="stringButton"      VALUE="Draw">
<PARAM NAME="stringResults"     VALUE="Contour values:">
<!- Other parameters here ->
</APPLET>
```

ContourPlotApplet.java

ContourPlotApplet is a container for all the user-interface elements. Its source code is shown in Listing 2. The class starts with a few final static variables (constants) the keyword final indicating that the variable's value may not be changed and the keyword static indicating that it is a class variable, not an instance variable. These are followed by the six user-interface components shown in **Figures 3**, **4** and **5**, and finally a number

MkLinux

Microkernel Linux for the Power Macintosh

Why wait for Rhapsody? MkLinux is a complete system, based on Linux 2 and the Mach 3 microkernel. It includes a complete software development system (gcc, gdb, perl, ...), X11R6.3, and hundreds of other commands. Get MkLinux and start working right now with Mach, Objective-C, UNIX development tools, and more...

MkLinux builds and runs most Intel-based Linux software. It works efficiently and reliably on a wide range of Power Macintosh platforms, with other ports on the way. MkLinux has an active user community and a substantial Reference Release, both sponsored by Apple Computer. If you want to get a jump start on Rhapsody, this is a great way to do it!

Visit Apple's MkLinux web site (www.mklinux.apple.com) and Prime Time Freeware's web site (www.ptf.com) to find out more about MkLinux and other fine freeware products.

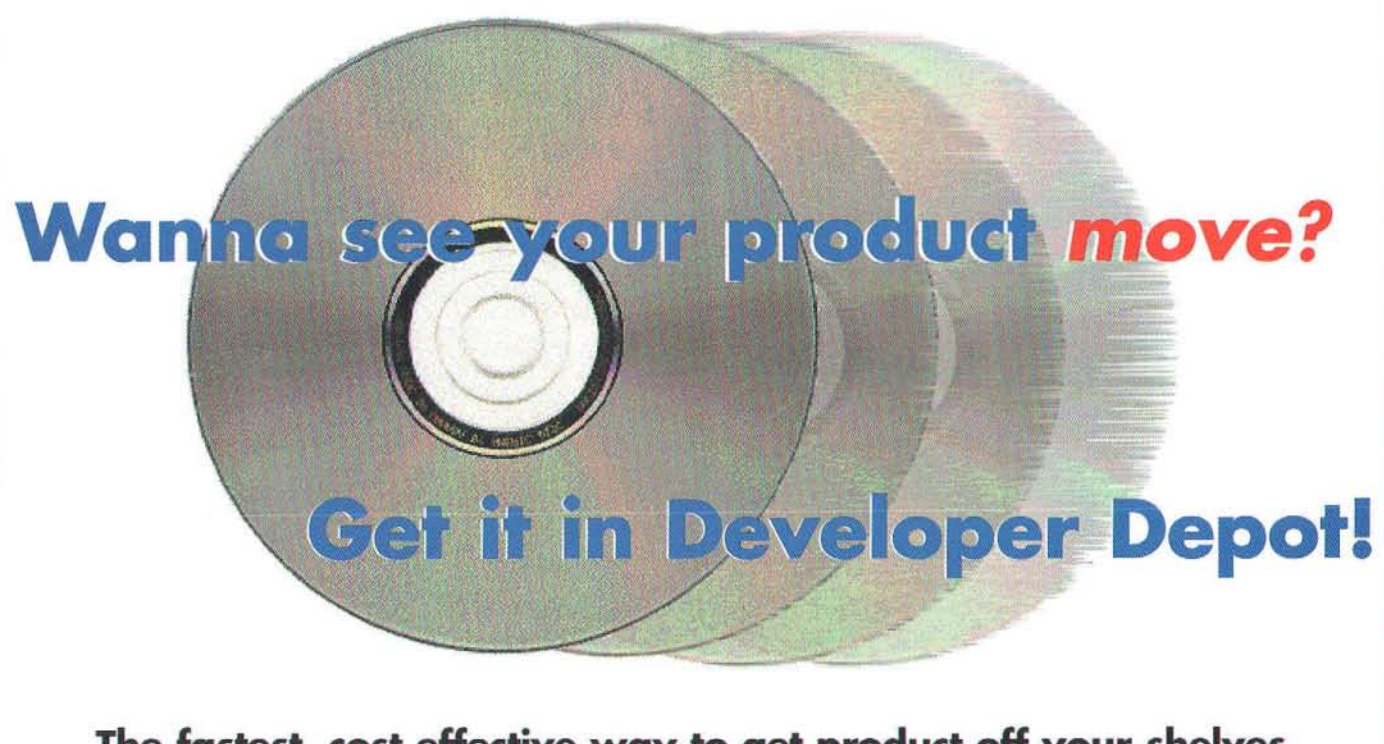
Prime Time Freeware	info@ptf.com
370 Altair Way, #150	(408) 433-9662
Sunnyvale, CA 94086	(408) 433-0727 fax

of static (class) variables which are String objects used to store strings read from the <APPLET> tag and subsequently to display messages in the results area. These data members are followed by the class' three methods init(), handleEvent(Event e) and DrawTheContourPlot() which are explained in the comments in the source code. This last method is the most important and calls several key methods in the ContourPlot object, especially thePlot.paint(Graphics g).

Listing 2

ContourPlotApplet.java

```
// "ContourPlotApplet" is the main class, that is, the applet,
// which is a container for all the user-interface elements.

import java.awt.*;
import java.io.*;

public class ContourPlotApplet extends java.applet.Applet {

  // Below, constants, i.e. "final static" data members:
  final static int  NUMBER_COMPONENTS = 6;
  final static int  MIN_X_STEPS=  2,
                    MIN_Y_STEPS=  2,
                    MAX_X_STEPS= 100,
                    MAX_Y_STEPS= 100;
  final static String EOL =
    System.getProperty("line.separator");
  final static String DEFAULT_Z  =
    "{{0.5,1.1,1.5,1,2.0,3,3,2,1,0.1},"  + EOL +
    " {1.0,1.5,3.0,5,6.0,2,1,1.2,1,4}," + EOL +
    " {0.9,2.0,2.1,3,6.0,7,3,2,1,1.4}," + EOL +
    " {1.0,1.5,3.0,4,6.0,5,2,1.5,1,2}," + EOL +
  " {0.8,2.0,3.0,3.4.0,4.3,2.4,2,3}," + EOL +
  " {0.6,1.1,1.5,1.4.0,3.5,3.2,3.4}," + EOL +
  " {1.0,1.5,3.0,5,6.0,2.1,1.2,2.7,4}," + EOL +
  " {0.8,2.0,3.0,3.5.5,6.3,2,1,1.4}," + EOL +
  " {1.0,1.5,3.0,4,6.0,5,2,1,0.5,0.2}}";

// Below, the six user-interface components:
ContourPlot thePlot =
  new ContourPlot(MIN_X_STEPS, MIN_Y_STEPS);
Label       zPrompt  = new Label("", Label.LEFT);
TextArea    zField   = new TextArea(DEFAULT_Z,30,6);
Checkbox    interBox = new Checkbox();
Button      drawBtn  = new Button();
TextArea    results  = new TextArea();

// Below, class data members read from the <APPLET> tag:
static String contourValuesTitle,infoStrX,infoStrY,
            errParse,errLog,errComp,errEqual,
            errExpect,errEOF,errBounds;

//------------------------------------------------------------------
// "init" overrides "super.init()" and initializes the applet by:
// 1.  getting parameters from the <APPLET> tag;
// 2.  setting layout to instance of "ContourPlotLayout";
// 3.  initializing and adding the six user-interface
//        components, using the method "add()" which will
//        also call "ContourPlotLayout.addLayoutComponent()".
//------------------------------------------------------------------
public void init() {
  infoStrX   = getParameter("stringX");
  infoStrY   = getParameter("stringY");

  setLayout(new ContourPlotLayout());
  add("thePlot", thePlot);
  zPrompt.setText(getParameter("stringZ"));
  add("zPrompt", zPrompt);
  zField.setBackground(Color.white);
  add("zField", zField);
  interBox.setLabel(getParameter("stringBox"));
  interBox.setState(false);
  add("interBox", interBox);
  drawBtn.setLabel(getParameter("stringButton"));
  drawBtn.setFont(new Font("Helvetica", Font.BOLD, 10));
  drawBtn.setBackground(Color.white);
  add("drawBtn", drawBtn);
  results.setEditable(false);
  results.setFont(new Font("Courier", Font.PLAIN, 9));
  results.setBackground(Color.white);
  add("results", results);
  contourValuesTitle = getParameter("stringResults");
  errParse    = getParameter("stringErrParse");
  errLog      = getParameter("stringErrLog1") + EOL +
                getParameter("stringErrLog2") + EOL +
                getParameter("stringErrLog3");
  errComp     = getParameter("stringErrComp");
  errEqual    = getParameter("stringErrEqual");
  errExpect   = getParameter("stringErrExpect");
  errEOF      = getParameter("stringErrEOF");
  errBounds   = getParameter("stringErrBounds");
}

//------------------------------------------------------------------
// Handle events. The only event not handled by the superclass
// is a mouse hit (i.e. "Event.ACTION_EVENT") in the "Draw" button.
//------------------------------------------------------------------
public boolean handleEvent(Event e) {
  if ((e != null) &&
     (e.id == Event.ACTION_EVENT) &&
     (e.target == drawBtn)) {
    DrawTheContourPlot();
    return true;
  }
  else return super.handleEvent(e);
}
```



The fastest, cost effective way to get product off your shelves.

For information: • Voice: 805/494-9797 • Fax: 805/494-9798 • E-mail: vendor_relations@devdepot.com

Order directly from web site at www.neologic.com and save $100!

Download the architectural overview or view it on-line with Adobe Acrobat 3.0

The most powerful object-oriented database engine available

NeoAccess® displays electrifying performance — up to ten times that of the competition. Behind its elegant programming interface is a fully optimized relational query engine built for speed. It also has an object cache with automatic garbage collection so your applications can run in a much smaller memory footprint than you ever imagined possible. And NeoAccess has no runtime fees so you pay one affordable price no matter how many copies of your application you use or sell.

While others promise cross-platform – NeoAccess delivers. NeoAccess is a set of C++ classes designed for use with popular compilers and frameworks on Windows®, Macintosh®, and Unix™ platforms. Thousands of developers, including America Online® and Claris®, have already found that NeoAccess enabled them to build fast, powerful internet applications in record time. That's why there are more NeoAccess based applications on end-user machines than any other object database backend.

M–F 9AM to 5PM Pacific:
1-800-919-6353
For Information and Customer Service:
1-510-524-5897

neo·logic®

NeoLogic Systems, Inc.
1450 Fourth St., Suite 12, Berkeley, CA 94710
v. 510.524.5897 f. 510.524.4501
neologic@neologic.com

```
//-----------------------------------------------------------
// "DrawTheContourPlot" does what its name says (in reaction to a hit on the
// "Draw" button). The guts of this method are in the "try" block which:
// 1. gets the interpolation flag (for contour values);
// 2. parses the data, i.e. the matrix of z values;
// 3. draws the contour plot by calling the "paint()"
//       method of the component "thePlot";
// 4. displays the results, i.e. the number of rows and columns in the grid,
//       an echo of the matrix of z values, and the list of contour values.
// This method catches 2 exceptions, then finally (i.e. regardless of exceptions)
// sends a completion message to the Java console using "System.out.println()".
//-----------------------------------------------------------
public void DrawTheContourPlot() {
   String   s;

   try {
      s = zField.getText();
      thePlot.logInterpolation = interBox.getState();
      thePlot.ParseZedMatrix(s);
      thePlot.paint(thePlot.getGraphics());
      s = thePlot.ReturnZedMatrix() +
         contourValuesTitle + EOL +
         thePlot.GetContourValuesString();
      results.setText(s);
   }
   catch(ParseMatrixException e) {
      thePlot.repaint();
      results.setText(e.getMessage());
   }
   catch(IOException e) {
      thePlot.repaint();
      results.setText(e.getMessage());
   }
   finally {
      System.out.println("Exiting DrawTheContourPlot");
   }
}
}
```

ContourPlotLayout.java

ContourPlotLayout is derived directly from java.lang.Object and implements the interface java.awt.LayoutManager. Its source code is shown in Listing 3. Recall that an "interface" in Java is an abstract class in which all methods are abstract, and is Java's limited way of implementing mix-in classes, that is, allowing a very restricted degree of multiple inheritance. Since an interface is completely abstract, all its methods must be overridden in any class which implements it, and that is the case here: ContourPlotLayout contains implementations of all five methods — addLayoutComponent, layoutContainer, minimumLayoutSize, preferredLayoutSize and removeLayoutComponent — declared abstractly in java.awt.LayoutManager.

The purpose of ContourPlotLayout is to lay out the six user-interface components inside our applet's rectangle. The Java API includes several layout managers, such as FlowLayout, BorderLayout, GridLayout, etc. (again, see Dave Mark's series of Getting Started articles in *MacTech*), but none was deemed appropriate here because it was desired to assign special fixed values to most (but not all) of the positions and dimensions of the components. The six components are stored in an instance variable, an array k, whose values k[0] through k[5] correspond to the applet's instance variables thePlot, zPrompt, zField, interBox, drawBtn and results. k[1] through k[4] are of fixed position and dimension. The other two components, that is, the contour plot and the results, also have fixed position, but may change in size as the applet's dimensions change (for example, if the applet's window is resized in the Java interpreter). The contour plot will be made as large as possible while remaining square, while its size never falls below a certain minimum, the constant (that is, static final) MIN_PLOT_DIMEN. The results area's width never changes, but its height expands to fill the available space while never falling below the minimum MIN_RES_HEIGHT.

Listing 3

ContourPlotLayout.java

```
// ContourPlotLayout implements the interface LayoutManager
// & is used by ContourPlotApplet to lay out its components.

import java.awt.*;
import java.io.*;

public class ContourPlotLayout
  extends     java.lang.Object
  implements java.awt.LayoutManager {

  // Below, constant data members:
  private static final int COUNT =
    ContourPlotApplet.NUMBER_COMPONENTS;
  private static final int
    MARGIN         =   5,
    MIN_PLOT_DIMEN = 300,
    LEFT_WIDTH     = 250,
    CBOX_WIDTH     = 130,
    BUTTON_H_POS   = MARGIN + CBOX_WIDTH + MARGIN,
    BUTTON_WIDTH   = LEFT_WIDTH - CBOX_WIDTH - MARGIN,
    LINE_HEIGHT    =  25,
    DATA_HEIGHT    = 105,
    MIN_RES_HEIGHT =  50,
    DATA_V_POS     = MARGIN + MARGIN + LINE_HEIGHT,
    BUTTON_V_POS   = DATA_V_POS + MARGIN + DATA_HEIGHT,
    RESULTS_V_POS  = BUTTON_V_POS + MARGIN + LINE_HEIGHT;

  // Below, data members: the array of components, the dimensions of
  // the contour  plot component and the height of the results area.
  Component  k[]  = new Component[COUNT];
  Dimension  d    = new Dimension( MIN_PLOT_DIMEN,
                                   MIN_PLOT_DIMEN);
  int results_height = MIN_RES_HEIGHT;

  //-----------------------------------------------------------
  // "addLayoutComponent" is necessary to override the
  // corresponding abstract method in "LayoutManager".
  //-----------------------------------------------------------
  public void addLayoutComponent(String name, Component c)
  {
    if (name.equals("thePlot")) {
      c.reshape( 2*MARGIN+LEFT_WIDTH, MARGIN,
                 d.width, d.height);
      addComponentNumber(0,c);
    }
    else if (name.equals("zPrompt")) {
      c.reshape( MARGIN, MARGIN,
                 LEFT_WIDTH, LINE_HEIGHT);
      addComponentNumber(1,c);
    }
    else if (name.equals("zField")) {
      c.reshape( MARGIN, DATA_V_POS,
                 LEFT_WIDTH, DATA_HEIGHT);
      addComponentNumber(2,c);
    }
    else if (name.equals("interBox")) {
      c.reshape( MARGIN, BUTTON_V_POS,
                 CBOX_WIDTH, LINE_HEIGHT);
      addComponentNumber(3,c);
    }
    else if (name.equals("drawBtn")) {
      c.reshape( BUTTON_H_POS, BUTTON_V_POS,
                 BUTTON_WIDTH, LINE_HEIGHT);
      addComponentNumber(4,c);
    }
    else if (name.equals("results")) {
      c.reshape( MARGIN, RESULTS_V_POS,
                 LEFT_WIDTH, results_height);
      addComponentNumber(5,c);
    }
//  throw new SomeKindOfException(
//    "Attempt to add an invalid component");
  }

  //-----------------------------------------------------------
  // "GetDimensions" computes the data members "d" and "results_height"
  // which are  the only dimensions in the layout which are not fixed.
  //-----------------------------------------------------------
  public void GetDimensions(Container parent) {
    d = parent.size();
    d.width = d.width - LEFT_WIDTH - 3*MARGIN;
    d.height = d.height - 2*MARGIN;
    if (d.width < MIN_PLOT_DIMEN)
      d.width = MIN_PLOT_DIMEN;
    if (d.height < MIN_PLOT_DIMEN)
      d.height = MIN_PLOT_DIMEN;
    if (d.width > d.height) d.width = d.height;
    else if (d.height > d.width) d.height = d.width;
    results_height = d.height + MARGIN - RESULTS_V_POS;
    if (results_height < MIN_RES_HEIGHT)
      results_height = MIN_RES_HEIGHT;
  }

  //-----------------------------------------------------------
  // "addComponentNumber" adds a component given its index
  // and is a utility routine used by "addLayoutComponent".
  //-----------------------------------------------------------
  public void addComponentNumber(int i, Component c) {
    if ((i < 0) || (i >= COUNT)) {
      throw new ArrayIndexOutOfBoundsException();
    }
    else if (k[i] != null) {
//  throw new SomeKindOfException(
//    "Attempt to add a component already added");
    }
    else k[i] = c;
  }

  //-----------------------------------------------------------
  // "layoutContainer" is necessary to override the
  // corresponding abstract method in "LayoutManager".
  //-----------------------------------------------------------
  public void layoutContainer(Container parent) {
    GetDimensions(parent);
    if (k[0] != null) k[0].reshape
      (2*MARGIN+LEFT_WIDTH,MARGIN,d.width,d.height);
    if (k[1] != null) k[1].reshape
      (MARGIN,MARGIN,LEFT_WIDTH,LINE_HEIGHT);
    if (k[2] != null) k[2].reshape
      (MARGIN,DATA_V_POS,LEFT_WIDTH,DATA_HEIGHT);
    if (k[3] !=null) k[3].reshape
      (MARGIN,BUTTON_V_POS,CBOX_WIDTH,LINE_HEIGHT);
    if (k[4] != null) k[4].reshape
      (BUTTON_H_POS,BUTTON_V_POS,
       BUTTON_WIDTH,LINE_HEIGHT);
    if (k[5] != null) k[5].reshape
      (MARGIN,RESULTS_V_POS,LEFT_WIDTH,results_height);
  }

  //-----------------------------------------------------------
  // "minimumLayoutSize" is necessary to override the
  // corresponding abstract method in "LayoutManager".
  //-----------------------------------------------------------
  public Dimension minimumLayoutSize(Container parent) {
    return new Dimension(
      3*MARGIN + LEFT_WIDTH + MIN_PLOT_DIMEN,
      2*MARGIN + MIN_PLOT_DIMEN);
  }

  //-----------------------------------------------------------
  // "preferredLayoutSize" is necessary to override the
  // corresponding abstract method in "LayoutManager".
  //-----------------------------------------------------------
  public Dimension preferredLayoutSize(Container parent) {
    GetDimensions(parent);
    return new Dimension( 3*MARGIN + d.width + LEFT_WIDTH,
                          2*MARGIN + d.height);
  }
  //———————————————————————————————————
  // "removeLayoutComponent" is necessary to override the
  // corresponding abstract method in "LayoutManager".
  //———————————————————————————————————
  public void removeLayoutComponent(Component c) {
    for (int i = 0; i < COUNT; i++)
      if (c == k[i]) k[i] = null;
  }
}
```

ContourPlot.java

ContourPlot, part of whose source code is shown in Listing 4, is derived from the class java.awt.Canvas. An instance of it is used by the applet as the user-interface component which parses the data, draws the contour plot, and returns a string of results. This class begins with a number of constants: note, for example, the characters OPEN_SUITE and CLOSE_SUITE specifying delimiters in the matrix to be parsed and BETWEEN_ARGS which specifies the data separator between values in the matrix; note also the platform-independent way of assigning a value to EOL, as recommended in [5].

The data members xSteps and ySteps are used to hold the number of horizontal and vertical steps, respectively, in the grid.

The matrix z will contain values of type float and is declared to have two indices but the number of components in each dimension is initially unspecified. The number of rows in z and the length of each row will be incremented as the data are read. The matrix will be made rectangular only after all data are parsed; in fact, Java syntax allows one to use an array of arrays (such as z here) in which the "inner" arrays need not have the same length. For example, the length of the x^{th} row of z is given by z[x].length. Notice that, according to standard Java practice, the memory allocated for the matrix z is never disposed, even when new data are parsed, since garbage collection is performed automatically by the Java interpreter. Or to express this in different words, the contents of z may be "disposed" by simply performing the assignment z = null which has the effect that any previous contents of z are no longer referenced (unless they have been assigned to some other variable other than z) and may thus be garbage-collected by the interpreter at its convenience.

The data members d, deltaX, deltaY are measurements, in pixels, of the dimensions of the contour plot and the distance between grid lines horizontally and vertically.

Most of the remaining data members are variables adapted from Snyder's Fortran subroutine GCONTR. See [2] for a discussion of their meaning.

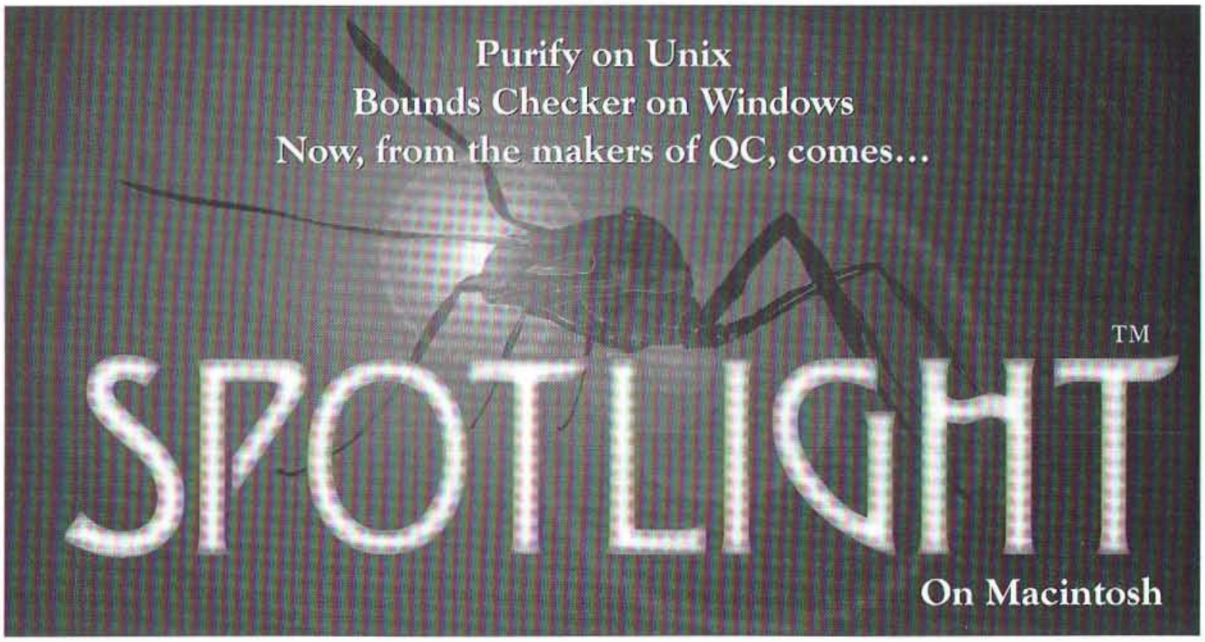
The various methods in the class ContourPlot are explained briefly by comments in the full source code included with the project. The most important are paint(Graphics g) and ContourPlotKernel(Graphics g, boolean workSpace[]). The former is called directly by the applet and in turn calls the latter which corresponds to the "outer" level of the plotting algorithm adapted from Synder's subroutine GCONTR. (For brevity, some methods have been omitted from Listing 4, in particular a few methods which are called directly or indirectly only by ContourPlotKernel and thus include only code adapted from Synder.)

Listing 4

Selections from ContourPlot.java

```
// "ContourPlot" is the most important class. It is a user-interface component which
// parses the data, draws the contour plot, and returns a string of results.

import java.awt.*;
import java.io.*;

public class ContourPlot extends Canvas {

  // Below, constant data members:
  final static boolean SHOW_NUMBERS = true;
  final static int  BLANK         = 32,
                    OPEN_SUITE    = (int)'{',
                    CLOSE_SUITE   = (int)'}',
                    BETWEEN_ARGS  = (int)',',
                    N_CONTOURS    = 10,
                    PLOT_MARGIN   = 20,
                    WEE_BIT       =  3,
                    NUMBER_LENGTH =  3;
  final static double   Z_MAX_MAX  = 1.0E+10,
                        Z_MIN_MIN  = -Z_MAX_MAX;
  final static String EOL =
    System.getProperty("line.separator");

  // Below, data members which store the grid steps,
  // the z values, the interpolation flag, the dimensions
  // of the contour plot and the increments in the grid:
  int       xSteps, ySteps;
  float     z[][];
  boolean   logInterpolation = false;
  Dimension d;
  double    deltaX, deltaY;

  // Below, data members, most of which are adapted from
  // Fortran variables in Snyder's code:
  int    ncv = N_CONTOURS;
  int    l1[] = new int[4];
  int    l2[] = new int[4];
  int    ij[] = new int[2];
  int    i1[] = new int[2];
  int    i2[] = new int[2];
  int    i3[] = new int[6];
  int    ibkey,icur,jcur,ii,jj,elle,ix,iedge,iflag,ni,ks;
  int    cntrIndex,prevIndex;
  int    idir,nxidir,k;
  double   z1,z2,cval,zMax,zMin;
  double   intersect[]  = new double[4];
  double   xy[]         = new double[2];
  double   prevXY[]     = new double[2];
  float    cv[]         = new float[ncv];
  boolean  jump;

  //———————————————————————————————————
  // A constructor method.
  //———————————————————————————————————
  public ContourPlot(int x, int y) {
    super();
    xSteps = x;
    ySteps = y;
    setForeground(Color.black);
    setBackground(Color.white);
  }

  //———————————————————————————————————
  // The following routines are omitted from this listing.
  // See the full source code included with the project.
  //
  // int sign(int a, int b)
  // void InvalidData()
  // void GetExtremes()
  // void SetMeasurements()
  // void DetectBoundary()
  // boolean Routine_label_020()
  // boolean Routine_label_050()
  // boolean Routine_label_150()
  // short Routine_label_200( Graphics g,
  //                                  boolean workSpace[])
  // void ContinueContour()
```

```
//---------------------------------------------------------------------
// "AssignContourValues" interpolates between "zMin" and "zMax", either
// logarithmically or linearly, in order to assign contour values to the array "cv".
//---------------------------------------------------------------------
void AssignContourValues() throws ParseMatrixException {
  int    i;
  double delta;

  if ((logInterpolation) && (zMin <= 0.0)) {
    InvalidData();
    throw new
      ParseMatrixException(ContourPlotApplet.errLog);
  }
  if (logInterpolation) {
    double temp = Math.log(zMin);

    delta = (Math.log(zMax)-temp) / ncv;
    for (i = 0; i < ncv; i++)
      cv[i] = (float)Math.exp(temp + (i+1)*delta);
  }
  else {
    delta = (zMax-zMin) / ncv;
    for (i = 0; i < ncv; i++)
      cv[i] = (float)(zMin + (i+1)*delta);
  }
}

//---------------------------------------------------------------------
// "GetContourValuesString" returns a list of the
// contour values for display in the results area.
//---------------------------------------------------------------------
String GetContourValuesString() {
  String s = new String();
  int    i;

  for (i = 0; i < ncv; i++)
    s = s + "[" + Integer.toString(i)
          + "] " + Float.toString(cv[i]) + EOL;
  return s;
}

//---------------------------------------------------------------------
// "DrawGrid" draws the rectangular grid of gray lines
// on top of which the contours will later be drawn.
//---------------------------------------------------------------------
void DrawGrid(Graphics g) {
  int i,j,kx,ky;

  // Interchange horizontal & vertical
  g.clearRect(0, 0,   d.height+2*PLOT_MARGIN,
                      d.width +2*PLOT_MARGIN);
  g.setColor(Color.gray);
  for (i = 0; i < xSteps; i++) {
    kx = (int)((float)i * deltaX);
    g.drawLine(PLOT_MARGIN,
               PLOT_MARGIN+kx,
               PLOT_MARGIN+d.height,
               PLOT_MARGIN+kx);
  }
  for (j = 0; j < ySteps; j++) {
    ky = (int)((float)j * deltaY);
    g.drawLine(PLOT_MARGIN+ky,
               PLOT_MARGIN,
               PLOT_MARGIN+ky,
               PLOT_MARGIN+d.width);
  }
  g.setColor(Color.black);
}

//---------------------------------------------------------------------
// "SetColour" sets the colour of the graphics object, given the contour
// index, by interpolating linearly between "Color.blue" & "Color.red".
//---------------------------------------------------------------------
void SetColour(Graphics g) {
  Color c = new Color(
    ((ncv-cntrIndex)  * Color.blue.getRed()   +
           cntrIndex  * Color.red.getRed())/ncv,
    ((ncv-cntrIndex)  * Color.blue.getGreen() +
           cntrIndex  * Color.red.getGreen())/ncv,
    ((ncv-cntrIndex)  * Color.blue.getBlue()  +
           cntrIndex  * Color.red.getBlue())/ncv);
  g.setColor(c);
}

//---------------------------------------------------------------------
// "DrawKernel" is the guts of drawing and is called directly or indirectly by
// "ContourPlotKernel" in order to draw a segment of a contour or to set the pen
// position "prevXY". Its action depends on "iflag":
//
// iflag == 1 means Continue a contour
// iflag == 2 means Start a contour at a boundary
// iflag == 3 means Start a contour not at a boundary
// iflag == 4 means Finish contour at a boundary
// iflag == 5 means Finish closed contour not at boundary
// iflag == 6 means Set pen position
//
// If the constant "SHOW_NUMBERS" is true, then the contour index is drawn
// adjacent to where the contour ends when completing a contour (iflag == 4 or 5).
//---------------------------------------------------------------------
void DrawKernel(Graphics g) {
  int prevU,prevV,u,v;

  if ((iflag == 1) || (iflag == 4) || (iflag == 5)) {
    if (cntrIndex != prevIndex) { // Must change colour
      SetColour(g);
      prevIndex = cntrIndex;
    }
    prevU = (int)((prevXY[0] - 1.0) * deltaX);
    prevV = (int)((prevXY[1] - 1.0) * deltaY);
    u = (int)((xy[0] - 1.0) * deltaX);
    v = (int)((xy[1] - 1.0) * deltaY);

    // Interchange horizontal & vertical
    g.drawLine( PLOT_MARGIN+prevV,PLOT_MARGIN+prevU,
                PLOT_MARGIN+v,  PLOT_MARGIN+u);
    if ((SHOW_NUMBERS) && ((iflag==4) || (iflag==5))) {
      if      (u == 0)        u = u - WEE_BIT;
      else if (u == d.width)  u = u + PLOT_MARGIN/2;
      else if (v == 0)        v = v - PLOT_MARGIN/2;
      else if (v == d.height) v = v + WEE_BIT;
      g.drawString(Integer.toString(cntrIndex),
        PLOT_MARGIN+v, PLOT_MARGIN+u);
    }
  }
  prevXY[0] = xy[0];
  prevXY[1] = xy[1];
}

//---------------------------------------------------------------------
// "CrossedByContour" is true iff the current segment inthe grid is crossed by
// one of the contour values and has not already been processed for that value.
//---------------------------------------------------------------------
boolean CrossedByContour(boolean workSpace[]) {
  ii = ij[0] + i1[elle];
  jj = ij[1] + i1[1-elle];
  z1 = z[ij[0]-1][ij[1]-1];
  z2 = z[ii-1][jj-1];
  for (cntrIndex = 0; cntrIndex < ncv; cntrIndex++) {
    int i = 2*(xSteps*(ySteps*cntrIndex+ij[1]-1)
               +ij[0]-1) + elle;

    if (!workSpace[i]) {
      float x = cv[cntrIndex];
      if ((x>Math.min(z1,z2)) && (x<=Math.max(z1,z2)))
      {
        workSpace[i] = true;
        return true;
      }
    }
  }
  return false;
}

//---------------------------------------------------------------------
// "ContourPlotKernel" is the guts of this class and
// corresponds to Synder's subroutine "GCONTR".
//---------------------------------------------------------------------
void ContourPlotKernel(Graphics g, boolean workSpace[])
{
  short val_label_200;

  l1[0] = xSteps;   l1[1] = ySteps;
  l1[2] = -1;       l1[3] = -1;
  i1[0] = 1; i1[1] = 0;
  i2[0] = 1; i2[1] = -1;
  i3[0] = 1; i3[1] = 0; i3[2] = 0;
  i3[3] = 1; i3[4] = 1; i3[5] = 0;
```


Purify on Unix
Bounds Checker on Windows
Now, from the makers of QC, comes...
SPOTLIGHT™
On Macintosh

Find Bugs Fast

Spotlight is the first **Automatic Memory Debugger** for the Macintosh. Instantly detect invalid memory accesses, bad toolbox parameters, leaks, stack overwrites, memory relocation problems, and much more.

Spotlight uses your XSYM file to automatically patch your application — no need to change your source code. No need to recompile. **No learning curve whatsoever.**

The interface gives you instant feedback when an error is detected. You can ignore the error, ignore all future occurrences of the error, or log the error to a text file for later analysis.

Fine Grained Memory Protection

Spotlight identifies reads and writes outside of the application and system heap. But it does not stop there. Spotlight's Object Code Replacement instrumentation **inspects each read and write instruction in your code,** detecting faulty memory reads and writes that occur between blocks, in released blocks, across multiple blocks, and in ROM.

This technology works on any heap object you can allocate: Mac heap objects, C 'malloc' heap objects, even C++ objects created with new.

Spotlight stops your application whenever a faulty memory access is about to occur and displays the **exact source code line** where it will happen. The calling stack is shown along with a display of all variables. A memory viewer shows the erroneous memory address and contents.

Toolbox Validation

Spotlight checks parameters to **over 400 toolbox API calls,** automatically validating handles, memory blocks, return values, and so on. Specific checks catch subtle errors such as drawing into an unlocked GWorld, passing an invalid window pointer, passing an address within an unlocked handle to a routine that may move memory, and too many more to list.

Leak Detection

No more struggling with MacsBug. On program exit Spotlight provides a **clear listing of all leaked memory** showing a full stack trace from where the memory was allocated. Instantly discover all leaked Mac OS objects, malloc'd objects, C++ objects, and resources.

Limitations

Spotlight requires an XSYM file to function. MPW and CodeWarrior users can generate these directly. Symantec users must use ToolServer to link with an XSYM capable linker. Spotlight requires a PowerPC processor.

Spotlight works on the object code. **No source code is required**: just the thing for testing purchased third party libraries.

Availability and Pricing

Pricing for Spotlight DR1 is $199 US (plus $5 shipping and handling within the continental US, $15 for international orders). This includes a free upgrade to the GM version and access to ftp interim upgrades leading up to GM. QC users can cross-grade to Spotlight for only $149.

All Onyx products carry a 30 day no questions asked money back guarantee.

Onyx Technology, Inc.
7811 27th Avenue West
Bradenton, Florida 34209

sales@onyx-tech.com www.onyx-tech.com
941 795-7801 941 795-5901 (fax)

Spotlight and QC are trademarks of Onyx Technology, Inc. All other trademarks are property of their respective owners.

Build great applications... Better Cheaper Faster!

Great | Good'nuf

Tools Plus

LIBRARIES + FRAMEWORK

Make cool apps while others are still reading their manuals. Professionally polished. Wickedly fast. Delightfully efficient.

"...the routines are more compact and faster than anything you might write...Every element of Tools Plus is useful... a bargain compared with coding these routines yourself." MW★★★★ Macworld

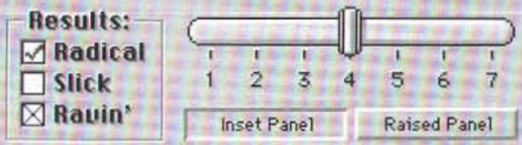
You build the interface.
Tools Plus provides the infrastructure.
It makes all your pieces work together as an application.
With only a few hundred routines, Tools Plus thins your code by tens of thousands of lines. You see results sooner.
Changes are a snap.

"All in all, it's an incredibly rich collection of tools... If you are interested in developing applications that have 'quality' written all over them, then Tools Plus is for you." MacTech MAGAZINE

Yes, Tools Plus has it!

- **Create any element** using a single routine
- **Everything works** as soon as you create it
- **Automates** all standard GUI elements
- **Windows**, **Tool bar** and **floating palettes**
- **Buttons** (all kinds, flat and 3D)
- **Scroll bars** (speed control, live scrolling)
- World-class **custom controls**
- **Fields** (w/scroll bars, filters, auto-editing)
- **Picture buttons** (the best anywhere)
- **List boxes**
- **CDEF** and **LDEF** automation
- **Cursors** (color, animated, auto-change)
- **Menus** (pull-down, hierarchical, pop-up)
- **Edit menu** (undo/redo, automatic editing)
- **Panels** (3D, group boxes, lines, panes)
- **3D titles** (raised or inset)
- Extended multi-monitor and color support
- **Clipboard** automation
- **Dynamic alerts** (no resources required)
- **Event processing** automation
- Over 500K of custom fonts, icons, cursors, and other useful resources

Plus thousands more exciting features and services!



FREE SuperCDEFs™ $89 value
✓ professionally designed and crafted controls
✓ dozens of 3D & flat buttons, tabs, sliders

Tools Plus for

Symantec (THINK) C/C++ *(68K)*	$149
THINK Pascal *(68K)*	$149
THINK C/C++ & THINK Pascal	$199
CodeWarrier Bronze *(68K)*	$199
CodeWarrier Gold *(68K and PowerPC)*	$249

(We accept VISA and Amex only. Add $10 for shipping.)
*Call for Academic pricing

Water's Edge Software
2441 Lakeshore Road West, Box 70022,
Oakville, Ontario Canada L6L 6M9

Orders and Enquiries:
Phone: (416)219-5628
Fax: (905)847-1638

WaterEdgSW@aol.com

Free Evaluation Kit:
Available at our web site
http://www.interlog.com/~wateredg

Water's Edge Software

```
    prevXY[0]   = 0.0; prevXY[1] = 0.0;
    xy[0]       = 1.0; xy[1] = 1.0;
    cntrIndex   = 0;
    prevIndex   = -1;
    iflag       = 6;
    DrawKernel(g);
    icur = Math.max(1,
      Math.min((int)Math.floor(xy[0]), xSteps));
    jcur = Math.max(1,
      Math.min((int)Math.floor(xy[1]), ySteps));
    ibkey = 0;
    ij[0] = icur;
    ij[1] = jcur;
    if (Routine_label_020() &&
        Routine_label_150()) return;
    if (Routine_label_050()) return;
    while (true) {
      DetectBoundary();
      if (jump)  {
        if (ix != 0)
          iflag = 4; // Finish contour at boundary
        iedge = ks + 2;
        if (iedge > 4) iedge = iedge - 4;
        intersect[iedge-1] = intersect[ks-1];
        val_label_200 = Routine_label_200(g,workSpace);
        if (val_label_200 == 1) {
          if (Routine_label_020() &&
              Routine_label_150()) return;
          if (Routine_label_050()) return;
          continue;
        }
        if (val_label_200 == 2) continue;
        return;
      }
      if ((ix != 3) && (ix+ibkey != 0) &&
        CrossedByContour(workSpace)) {
        //
        // An acceptable line segment has been found.
        // Follow contour until it hits a boundary or closes.
        //
        iedge = elle + 1;
        cval = cv[cntrIndex];
        if (ix != 1) iedge = iedge + 2;
        iflag = 2 + ibkey;
        intersect[iedge-1] = (cval - z1) / (z2 - z1);
        val_label_200 = Routine_label_200(g,workSpace);
        if (val_label_200 == 1) {
          if (Routine_label_020() &&
              Routine_label_150()) return;
          if (Routine_label_050()) return;
          continue;
        }
        if (val_label_200 == 2) continue;
        return;
      }
      if (++elle > 1) {
        elle = idir % 2;
        ij[elle] = sign(ij[elle],l1[k-1]);
        if (Routine_label_150()) return;
      }
      if (Routine_label_050()) return;
    }
}

//-----------------------------------------------------------
// "paint" overrides the superclass' "paint()" method. This method draws the grid and
// then the contours, provided that the first two contour values are not equal
// (which would indicate invalid data). The "workSpace" is used to remember which
// segments in the grid have been crossed by which contours.
//-----------------------------------------------------------
public void paint(Graphics g)
{
    int     workLength = 2 * xSteps * ySteps * ncv;
    boolean  workSpace[]; // Allocate below if data valid

    SetMeasurements();
    DrawGrid(g);
    if (cv[0] != cv[1]) { // Valid data
      workSpace = new boolean[workLength];
      ContourPlotKernel(g, workSpace);
    }
}
```

```
//------------------------------------------------------
// "ParseZedMatrix" parses the matrix of z values
// which it expects to find in the string "s".
//------------------------------------------------------
public void ParseZedMatrix(String s)
  throws ParseMatrixException, IOException
{
  StringBufferInputStream i;
  StreamTokenizer      t;

  i = new StringBufferInputStream(s);
  t = new StreamTokenizer(i);

  z = null; // Junk any existing matrix
  EatCharacter(t.OPEN_SUITE);
  do ParseRowVector(t);
  while (t.nextToken() == BETWEEN_ARGS);
  if (t.ttype != CLOSE_SUITE) {
    InvalidData();
    throw new ParseMatrixException(
      ContourPlotApplet.errParse + EOL +
      ContourPlotApplet.errExpect+(char)CLOSE_SUITE);
  }
  if (t.nextToken() != t.TT_EOF) {
    InvalidData();
    throw new ParseMatrixException(
      ContourPlotApplet.errParse + EOL +
      ContourPlotApplet.errEOF);
  }
  MakeMatrixRectangular();
  GetExtremes();
  if (zMax > Z_MAX_MAX) zMax = Z_MAX_MAX;
  if (zMin < Z_MIN_MIN) zMin = Z_MIN_MIN;
  AssignContourValues();
}

//------------------------------------------------------
// "ParseRowVector" parses a row of data from the stream.
//------------------------------------------------------
public void ParseRowVector(StreamTokenizer t)
  throws ParseMatrixException, IOException
{ // Parse a row of float's and
  // insert them in a new row of z[][]
  if (z == null) z = new float[1][];
  else AddRow();
  EatCharacter(t.OPEN_SUITE);
  do {
    if (t.nextToken() == t.TT_NUMBER) {
      int x = z.length - 1;

      if (z[x] == null) {
        z[x] = new float[1];
        z[x][0] = (float)t.nval;
      }
      else AddColumn((float)t.nval);
    }
    else {
      int x = z.length - 1;
      int y = z[x].length - 1;

      InvalidData();
      throw new ParseMatrixException(
        ContourPlotApplet.errParse + EOL +
        ContourPlotApplet.errComp + " [" +
        Integer.toString(x) + "," +
        Integer.toString(y) + "]");
    }
  } while (t.nextToken() == BETWEEN_ARGS);
  if (t.ttype != CLOSE_SUITE) {
    InvalidData();
    throw new ParseMatrixException(
      ContourPlotApplet.errParse + EOL +
      ContourPlotApplet.errExpect+(char)CLOSE_SUITE);
  }
}

//------------------------------------------------------
// "AddRow" appends a new empty row to the end of "z"
//------------------------------------------------------
public void AddRow() throws ParseMatrixException {
  int leng = z.length;
  float temp[][];

  if (leng >= ContourPlotApplet.MAX_X_STEPS)
    throw new ParseMatrixException(
      ContourPlotApplet.errParse + EOL +
      ContourPlotApplet.errBounds);
  temp = new float[leng+1][];
  System.arraycopy(z, 0, temp, 0, leng);
  z = temp;
}

//------------------------------------------------------
// "AddColumn" appends "val" to end of last row in "z"
//------------------------------------------------------
public void AddColumn(float val)
  throws ParseMatrixException
{
  int i = z.length - 1;
  int leng = z[i].length;
  float temp[];

  if (leng >= ContourPlotApplet.MAX_Y_STEPS)
    throw new ParseMatrixException(
      ContourPlotApplet.errParse + EOL +
      ContourPlotApplet.errBounds);
  temp = new float[leng+1];
  System.arraycopy(z[i], 0, temp, 0, leng);
  temp[leng] = val;
  z[i] = temp;
}

//------------------------------------------------------
// "MakeMatrixRectangular" appends zero(s) to the end of
// any row of "z" which is shorter than the longest row.
//------------------------------------------------------
public void MakeMatrixRectangular() {
  int i,y,leng;

  xSteps = z.length;
  ySteps = ContourPlotApplet.MIN_Y_STEPS;
  for (i = 0; i < xSteps; i++) {
    y = z[i].length;
    if (ySteps < y) ySteps = y;
  }
  for (i = 0; i < xSteps; i++) {
    leng = z[i].length;
    if (leng < ySteps) {
      float temp[] = new float[ySteps];

      System.arraycopy(z[i], 0, temp, 0, leng);
      while (leng < ySteps) temp[leng++] = 0;
      z[i] = temp;
    }
  }
}

//------------------------------------------------------
// "ReturnZedMatrix" returns a string containing the
// values in "z" for display in the results area.
//------------------------------------------------------
public String ReturnZedMatrix() {
  String s,oneValue;
  int    i,j;
  s = new String(
    ContourPlotApplet.infoStrX + xSteps + EOL +
    ContourPlotApplet.infoStrY + ySteps + EOL);
  for (i = 0; i < xSteps; i++) {
    for (j = 0; j < ySteps; j++) {
      oneValue = Double.toString(z[i][j]);
      while (oneValue.length() < NUMBER_LENGTH)
        oneValue = " " + oneValue;
      s = s + oneValue;
      if (j < ySteps-1) s = s + " ";
    }
    s = s + EOL;
  }
  return s;
}
```

```
  //-----------------------------------------
  // "EatCharacter" skips any BLANK's in the stream and
  // expects the character "c", throwing an exception if
  // the next non-BLANK character is not "c".
  //-----------------------------------------
  public void EatCharacter(StreamTokenizer t, int c)
    throws ParseMatrixException, IOException
  {
    while (t.nextToken() == BLANK) ;
    if (t.ttype != c) {
      InvalidData();
      throw new ParseMatrixException(
        ContourPlotApplet.errParse + EOL +
        ContourPlotApplet.errExpect + (char)c);
    }
  }
}
```

ParseMatrixException.java

ParseMatrixException, a very small class whose source code is shown in Listing 5, extends java.lang.Exception and is used to throw exceptions when any error occurs during parsing of the matrix of z values. It contains no new data members and its only method is a constructor taking a single String argument whose contents explain the error. The applet catches this exception and displays the string in the results box. The various explanatory strings are built from arguments read from the <APPLET> tag in the html file and stored as static String objects in ContourPlotApplet.

Listing 5

ParseMatrixException.java

```
// Class "ParseMatrixException" is used to signal an error corresponding to invalid
// data encounteredwhen parsing the matrix of z values.

public class ParseMatrixException extends Exception {

  public ParseMatrixException(String message) {
    super(message);
  }
}
```

Conclusion

This article has presented a reasonably simple Java applet which nevertheless performs a useful function. It illustrates a variety of features of the Java language, such as:

- constants, that is, final static data members;
- class (static) methods (See, for example, Float.toString called by GetContourValuesString or Math.log() called by AssignContourValues, in Listing 4.);
- manipulation of characters strings using the String object;
- parsing data by breaking it into tokens (see ParseZedMatrix() in Listing 4);
- sending output to the Java console (see DrawTheContourPlot in Listing 2);
- several user-interface components (see the data members in Listing 2);
- interfaces and a custom layout (see Listing 3);
- a custom component (see Listing 4);
- arrays of one or two dimensions (see, for example, data member z in Listing 4);
- applet parameters (see Listing 1 and init() in Listing 2);
- a little colour (see DrawGrid and SetColour in Listing 4);
- throwing and catching exceptions (see ParseZedMatrix() in Listing 4 and DrawTheContourPlot in Listing 2); custom exceptions (see Listing 5); etc.

For the reader who would like to experiment with possible improvements to this applet, here are a few suggestions:

- allow user-input of the number of contour values, or of the contour values themselves;
- implement a file dialogue so the user can read a matrix of data from a disk file;
- shade the regions between contours;
- allow the option of keeping grid cells square when the number of rows does not equal the number of columns — thus requiring a non-rectangular drawing area;
- allow user-input of the grid values — i.e. x and y values — so that the grid lines need not be equally spaced.
- for the ambitious: parse a closed-form expression such as z = sin(x y), then generate the grid values — choosing the fineness of the grid according to the absolute values of the partial derivatives of z — and finally plot the result;
- again, for the ambitious: implement Preusser's algorithm, for nice smooth curves!

The contour plotting applet (not necessarily the version described in this article, but similar) may be viewed by pointing your web browser to <http://www.CRM.UMontreal.CA/Galerie/ContourPlotApplet_Eng.html>.

References

[1] Dave Mark, "Java Break," MacTech Magazine, 12, 5 (May 1996), 7-12. (and subsequent months)
[2] W. V. Snyder, "Algorithm 531, Contour plotting [J6]," ACM Trans. Math. Softw. 4, 3 (Sept. 1978), 290-294.
[3] A. Preusser, "Algorithm 671, FARB-E-2D: Fill Area with Bicubics on Rectangles—A Contour Plot Program," ACM Trans. Math. Softw. 15, 1 (March 1989), 79-89.
[4] D. Flanagan, Java in a Nutshell, O'Reilly & Associates (1996).
[5] 100% Pure Java Cookbook, Version 5.01.97, Sun Microsystems (1997).

MT

If you want to be in the know, then you need <u>every</u> article published in the first 12 years of MacTech® Magazine and Apple's develop™ issues 1-29!

...in THINK Reference™ format!

So hurry, pick up the phone, fire up the e-mail, launch that fax machine, or simply drop by our web site and order yourself this new release of the MacTech® CD-ROM.

- Almost 1600 articles from all 139 issues of MacTech Magazine (1984-1996)
- Includes Apple develop issues 1-29
- Improved hypertext, improved indices, and a new THINK Reference Viewer — for lightning quick access!
- New hyperlinks between articles
- 100+ MB of source code — use them in your applications, with no royalties!
- Full version of THINK Reference — the original online guide to Inside Macintosh, Vols. I-VI
- 80MB of FrameWorks/SFA archives and the most complete set of FrameWorks archives known
- Sprocket™! MacTech's tiny framework that compiles quickly and supports System 7.5 features
- The best threads from the Mac programmer newsgroups plus thousands of notes, tips, snippets, and gotchas
- Popular tools that Mac programmers use to increase their productivity and much more!

Developer DEPOT™

Web Site: http//www.devdepot.com • E-mail: orders@devdepot.com
Phone: 800-MACDEV-1 • Outside the U.S. & Canada: 805-494-9797 • Fax: 805-494-9798

by Walt Brainerd, David Epstein and Richard Hendrickson

The F Programming Language Tastes Like Java

F is a new programming language derived from Fortran in much the way Java is derived from C

INTRODUCTION

This article describes the startling comeback of Fortran by discussing the features found in the F subset of Fortran 95 comparing them with the features found in Java. The combined strengths of Java and the F programming language make a wonderful introduction for beginners and a powerful package for professional programmers working on large projects. Even people familiar with modern Fortran and Java are unlikely to have noticed the benefits of combining the two.

Both languages claim multi-platform portability. Java capitalizes on the look and feel of C++ while aiming at professional programmers; F capitalizes on the efficiency and pedagogical design of Fortran 95 while aiming at teenage beginners, experienced scientists, engineers and large teams of programmers. Java offers powerful object-oriented capabilities familiar to the C++ community; F offers organized module-oriented capabilities familiar to the Fortran 95 community.

Getting over the initial shock that Fortran is not all that different from Java can be a stretch for any programmer not in a coma since 1995. The amazing similarities surprised even the authors, who had brought together over four decades of Fortran language design committee experience to create F while the Green team was creating Java.

The simple step taken by both language design teams is a move away from procedures as a first class citizen and the insistence that procedures be encapsulated inside a more powerful mechanism — a class in Java and a module in F.

After comparing the features of F and Java, we suggest some benefits of combining F and Java for introductory programming instruction as well as for large professional projects and efficiency driven applications.

A BINARY TREE

An example may help to show the similarities between F and Java. Listing 1 shows a binary tree written in both F and Java. Notice that the mythical phrase, "Java does not have pointers," is better said as, "Java does not support the dangerous pointer arithmetic found in C and C++."

The tree is made possible by the below statements that declare a recursive type:

```
type (node), pointer :: left, right ! found inside F "node"
  Tree left, right;          // found inside Java "Tree"
```

In F, the word "pointer" is actually used in the declaration of the "nested defined type;" in Java, the word "pointer" is not used in the declaration of the "nested class."

Walt Brainerd <walt@imagine1.com> is co-author of about a dozen programming books. He has been involved in Fortran development and standardization for over 25 years and was Director of Technical Work for the Fortran 90 standard.

David Epstein <david@imagine1.com> is the project editor of Part 3 of the Fortran standard regarding conditional compilation. He is the developer of the Expression Validation Test Suite (EVT) for Fortran compilers and author of the book Introduction to Programming with F.

Dick Hendrickson <dick@imagine1.com> has worked on Fortran compiler development in both educational and industrial environments since 1963. He currently is a consultant on compiler optimization and one of the developers of SHAPE, a test suite for Fortran compilers.

Listing 1: BinTree.f and BinTree.j

BinaryTree

A binary tree written in F is easily translated into Java.

```
!A Binary Tree in F                          //A Binary Tree in Java
module m_binary_tree                           class Tree {
 public  :: main, Insert, PrintTree
 private :: NewTree
 type, public :: node
  integer :: value                               int value;
  type (node), pointer :: left, right            Tree left, right;
 endtype node
contains
  function NewTree(num) result(tree)             Tree (
   integer, intent(in) :: num                     int num) {
   type (node), pointer :: tree
   allocate (tree)
   tree%value = num                                value = num;
   nullify(tree%left)                              left = null;
   nullify(tree%right)                             right = null;
  endfunction NewTree                            }//Constructor

  recursive subroutine Insert(tree, num)           static Tree Insert(
   type (node), pointer :: tree                   Tree tree,
   integer, intent(in) :: num                       int num) {
   if (.not. associated (tree)) then               if (tree==null) {
     tree = NewTree(num)                           return new Tree(num);
   elseif (num < tree%value) then                  }elseif(num
<tree.value){
     call Insert(tree%left)                          tree.left = Insert
                                                     (tree.left, num);
                                                     return tree;
   else                                            } else {
     call Insert(tree%right)
tree.right = Insert
                                                     (tree.right, num);
                                                     return tree;
   endif                                           }//if
  endsubroutine Insert                           }//Insert

  recursive subroutine PrintTree(tree)             static void
PrintTree(
   type (node), pointer :: tree                   Tree tree) {
   if (associated (tree)) then                      if (tree != null) {
     call PrintTree (tree%left)
PrintTree(tree.left);
     print *, tree%value                             System.out.println
                                                     (tree.value);

     call PrintTree (t%right)
PrintTree(tree.right);
   endif                                           }//if
  endsubroutine PrintTree                        }//PrintTree

  subroutine main                                public static void
                                               main (String argv[]) {
   type (node), pointer :: tree                    Tree tree;
   integer :: i                                    int i;
   nullify (tree)                                  tree = null;
   print *, "Unsorted list"                          System.out.println
                                                   ("Unsorted list");
   do i = 1, 30, 3                             For (i=1; i<3*10; i=i+3){
     print *, modulo(i,10)                           System.out.println
                                                     (i%10);
     call insert(tree, modulo(i,10))                 tree = Insert
                                                         (tree, i%10);
   enddo                                           }//for
   print *, "Sorted list"                          System.out.println
                                                   ("Sorted list");
   call print_tree (tree)                          PrintTree(tree);
  endsubroutine main                             }//main
endmodule m_tree_sort                          }//Tree

program tree_sort
  use m_tree_sort
  call main()
endprogram tree_sort
```

The strategy for pointers in F and Java is actually the same — no physical address is made available and what is being pointed to must be specified in the declaration (that is, no void pointers). In F, pointers only can point to objects that are targets; in Java, everything is being pointed to, even procedures. A non-mainstream view of Java is not that there are no pointers, but that everything in Java is being pointed to.

The F Programming Language

We expect the average reader to be more familiar with Java than with the F programming language. Thus, an introduction to some of the F programming language is called for.

One major difference between F and Java is that Java technology often refers to much more than a programming language — the class hierarchy from which all objects are derived, the Java Virtual Machine (JVM), bytecode and Just-In-Time compilers, to name just a few. F, on the other hand, does not have a standard module hierarchy or set of software tools considered part of the F technology other than the F compilers. Thus, this overview of F is a description of the F programming language. For those seeking more specifics, the grammar for F (in BNF) can be found on the Imagine1 web page (listed at the end of this article).

F Statements

Figure 1 categorizes all the F statements. Instead of calling a procedure main as in Java, there is a single main program required in F and it can be given any name. An F program can use modules and reference public procedures and other public entities from the used modules. Modules and their contained procedures can also use other modules. This creates module hierarchies whose roots are the modules that were used in the program. Using a module is similar to importing a package in Java. Both languages encourage a hierarchical organization.

```
Organizational Constructs
 program
 use <module>...  module
 endprogram         public  :: proc-list
                    private :: proc-list
                  contains
                    <procs>........subroutine........function
                    endmodule      use module      use module
                    endsubroutine  endfunction

Action Constructs
 if / elseif / else / endif
 select case / case / case default / endselect
 do / cycle / exit / enddo
 where / elsewhere / endwhere

Declarations
 type     integer  character    intrinsic  interface
 endtype  real     logical      module     procedure
          complex                          endinterface

Actions
 =  (assignment)          allocate     call     stop
 => (pointer assignment)  deallocate   return

Input/Output
 print  open  write   inquire   backspace
 read   close                   rewind
                                endfile
```

Figure 1. *Categorizing all the F statements.*

Mac, go out and touch the world for only $199!

ADB I/O lets your customers' Macs control things, it lets them feel.
ADB I/O lets the Mac be part of the physical world.

Thousands of Uses

Science, Multimedia, Children's Museums, Home Automation, Theatre Stages, Industrial Testing, Medical Research, Bonsai Watering, Robotics, Weather Stations—anything that can be electronically measured or controlled can use the ADB I/O.

No Serial Ports Occupied

ADB I/O uses the Apple Desktop Bus to communicate inputs and outputs to and from your Macintosh. (Maximum polling frequency is 90 Hz.) No external power supply is needed.

Eight I/O Channels Provided

Four relays for output. Four channels for Digital In, Digital Out or 8-bit Analog In.

Extensive Software Support

With ADB I/O and nearly any environment,* it is easy to build customized applications for your control and data acquisition needs.
For more info, visit us at
www.bzzzzzz.com.

BEEHIVE
TECHNOLOGIES INC.
The High-Tech Cottage Industry Cooperative On the Internet

*ADB I/O supports three language environments, two scripting environments, multimedia development environments, and other specific application environments with more on the way. Visit our home page to find out more and to download the complete manual as well as all supporting software at: **http://www.bzzzzzz.com.**
(818) 304-0664 voice. e-mail: contact@mail.bzzzzzz.com (818) 568-1530 fax. © 1997 BeeHive Technologies, Inc. All rights reserved.

One design goal in F is to require explicit declarations for entities and not rely on any defaults. This is the opposite extreme from the "old Fortran" with its implicit declaration of an integer by starting its name with the letter "i" as in

```
itotal = 0 ! Declares an integer in earlier Fortrans
```

One result of requiring everything to be explicitly stated is that the student's first module that contains a procedure will require a public statement. This leads into an early discussion of public and private if the instructor so chooses.

Those challenged to make a thorough comparison of F and Java will notice that Java does allow defaults and that the default for a function is protected. If Java were to employ the strategy used in F, the word protected would have to be specified explicitly.

The F Attributes

Figure 2 lists the attributes for F entities. A little description of public and private is given here as these attributes differ slightly from the others.

Attribute	Description
pointer	target Pointers can only point at objects that are targets.
public	private All module entities are either public or private.
intent	All subroutine arguments must be declared as intent in, out, or inout. All function arguments must be intent in.
dimension	Arrays can be from one to seven dimensions.
allocatable	For dynamic allocation.
parameter	A constant.
save	A static local variable.

Figure 2. *Attributes for F entities.*

Procedures are listed in public or private statements in the module that defines them. All other module entities require a public or private attribute in their declaration statement. One other use of the word "private" is to declare the components of a public defined type as private to the module. Such a module would supply its users with access to the defined type and the procedures that operate on entities declared to be of this type. The insides of the type are hidden from the users of this module. An example of this can be seen in Listing 2, which hides the internal parts of a car from its users.

Listing 2: CarModule.f

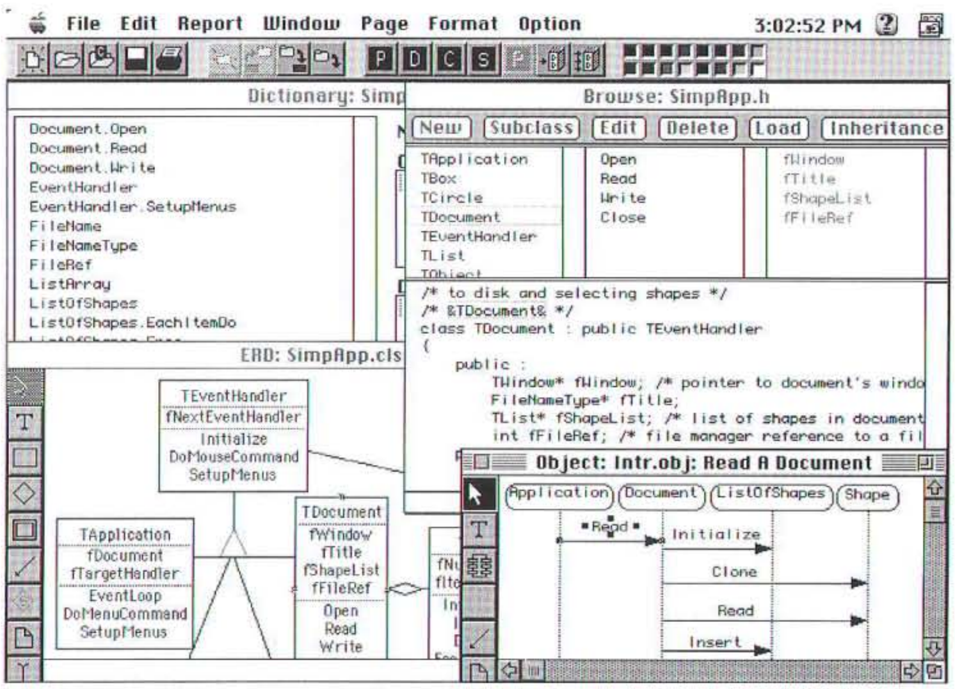
CarModule

This car module exports a type that has private components.

```
module m_car
use m_engine
 private ! do not export entities from module m_engine
 public :: GetCarInfo !, ... public procs listed here
 private :: IncreaseCost !, ... private procs listed here
 type, public :: t_car
 private ! components are private to this module
 type (t_engine) :: engine ! t_engine from m_engine
 integer :: cost_of_engine
 integer :: year
 endtype t_car
contains
 subroutine GetCarInfo(engine, year)
 ! ...
 subroutine IncreaseCost()
 ! ...
endmodule m_car
```

The Fundamental Difference Between F and Java

Although there are many similarities between F and Java, there is one fundamental design difference and a few minor differences that deserve attention.



```
Document.Open
Document.Read
Document.Write
EventHandler
EventHandler.SetupMenus
FileName
FileNameType
FileRef
ListArray
ListOfShapes
ListOfShapes.EachItemDo
```

```
TApplication    Open     fWindow
TBox            Read     fTitle
TCircle         Write    fShapeList
TDocument       Close    fFileRef
TEventHandler
TList
TObject
```

```
/* to disk and selecting shapes */
/* &TDocument& */
class TDocument : public TEventHandler
{
    public :
        TWindow* fWindow; /* pointer to document's windo
        FileNameType* fTitle;
        TList* fShapeList; /* list of shapes in document
        int fFileRef; /* file manager reference to a fil
```

The sporty new interface in MacA&D 6.0 enables designers to click to properties, dictionary details, integrated specifications, code or related requirements for any selected diagram object.

See http://www.excelsoftware.com for more details.

Software Engineering Made Easy!

MacA&D and WinA&D simplify system analysis, track requirements specifications, automate popular modeling techniques and generate code from your design. Throughout the process, extensive verification reports catch errors early to help you produce a quality product.

Powerful capabilities are just one advantage. Years of innovation have made these products easy to use and each includes extensive examples and tutorials. MacA&D and WinA&D share documents so your project team can use Macintosh, Solaris, HP-UX or Windows 95/NT computers. Select from OMT, Booch, Shlaer/Mellor, Coad/Yourdon, Fusion or Jacobson methods or pick and mix the best of each.

- Structured Analysis & Design
- Object-Oriented Analysis & Design
- Data, State and Task Modeling
- Integrated Code Editing & Browsing
- Multi-User Dictionary
- Requirements/Use Cases
- Generates C, C++, Delphi, SQL, etc.

Powerful Tools For Software Design

Powerful integration of system analysis, software design and code generation
Software Engineering
Made Easy!
Excel Software
MacA&D
for Macintosh

MacA&D
for Macintosh, Solaris and HP-UX

Powerful integration of system analysis, software design and code generation
Software Engineering
Made Easy!
Excel Software
WinA&D

WinA&D
for Windows 95/NT

See our web site or drop us an email to receive detailed technical brochures for MacA&D or WinA&D. Check out the new MacTranslator and WinTranslator products to generate design documents automatically from existing source code.

Excel Software
515-752-5359

The fundamental difference between F and Java is the way in which modules and classes are accessed. In F, a module can be used, which makes its public entities available. There are two sorts of modules worth mentioning for this discussion:

1. Declare actual data entities to be shared amongst its users.
2. Provide the definition of types and procedures that operate on those types, requiring the users to declare the actual data entities, and call procedures accordingly.

In Java, a class can be imported, making its definition available for instantiations. This is similar to the second sort of F module listed above. Namely, the Java class definition provides a type and procedures that operate on that type, but it is left to the importer of this class to create instantiations and call procedures accordingly.

Although modules are similar to classes, the fact that modules are used instead of instantiated leads to a slight shift in organizational thinking. For example, if SetSalary is a procedure that sets the salary of an employee, and employee is part of a person, a Java reference may look like

```
anEmployee.SetSalary(100);
and an F reference may look like
call SetSalary(anEmployee, 100)
```

In Java, an Employee would automatically inherit the features of a person; in F, it is up to the designer of the employee module to use the person module and make appropriate features of a person available to employees. For example, if Gender is a feature of a person, a Java reference may look like

```
anEmployee.GetGender();
and an F reference may look like
GetEmployeeGender(anEmployee)
```

Name Space

F references do not use dot notation. This presents a possible name space problem because the same name cannot be public in more than one used module within the scope of the use statements. The solution for this in F is the feature to rename used entities. For example,

```
use employee_module, s1=>salary
```

makes the salary entity of the employee_module available only as the name s1 in the scope of this use statement.

Cats, Dogs, and Mammals

A more philosophical view of comparing module-oriented programming and object-oriented programming can be shown using the example of cats and dogs. One argument for object-oriented programming is that humans see the world in terms of objects instead of in terms of actions. We see cats and dogs instead of GiveBath, HuntBird, WagTail and SniffHand. This argument also holds for module-oriented programming with the word "module" replacing the word "object" — humans see the world in terms of modules instead of in term of actions. Seeing cats and dogs as modules means viewing them as concepts which have definitions; it is possible to have an actual cat or dog object, but it is also possible to talk about them without having to create an instance of cat or dog.

The implementation difference can be found by taking a closer look at the writing and accessing of cats and dogs. In F, the implementor of cats and dogs can decide to use mammals to gain access to concepts common to all mammals. An F cat module cannot, however, export the mammal public entities. If the cat module wants to export a trait common to mammals such as SuckMilk, the implementor of the cat module must do some work to make this happen. In Java, when the cat class inherits from the mammal class, the cat class automatically exports all the public concepts of the mammal class. The access to cats in F is done by using the cat module and possibly calling KittenSucksMilk(aCat). The access to cats in Java is done by instantiating a cat and possibly calling aCat.SuckMilk.

Both methods have their advantages and disadvantages. Do you see cats and dogs or do you see mammals that are cats and dogs? In F, the designer of cats and dogs decides if it is important that they are mammals; in Java, the user of cats and dogs must understand that they are mammals.

Other Differences Between F and Java

Having hurdled the fundamental difference of using modules versus instantiating classes, other differences are minor jumps by comparison.

1. Intrinsic Procedures Versus Standard Libraries
2. F provides more than one hundred intrinsic procedures such as cos, tan, len and date_and_time. These intrinsic procedures are part of the F programming language and not part of a standard module that must be included or is implicitly included. Java provides much more than intrinsic procedures with its class hierarchy providing "standard" libraries for GUI building, exception handling, graphics, networking, multi-threading and coercion (such as integers to characters). This Java library will surely expand for whatever else becomes important (such as multimedia). To reference these goodies, the appropriate classes or packages must be imported. A wildcard of "*" can be used to import everything at the expense of size and speed of the application.

Overloaded Operators and Overloading Procedures

Both F and Java support using the same generic name to reference one of a group of specific procedures. Listing 3 shows overloading the swap for the F intrinsic types. Overloading operators is a different story.

Listing 3: Swap.f

SwapModule

This module overloads the name "swap" for all the intrinsic types.

```
module swap_module
 public :: swap

 interface swap
  module procedure int_swap, real_swap, complex_swap, &
           logical_swap, character_swap
 end interface !swap

contains
```

```
subroutine int_swap (a,b)
 integer, intent(in out) :: a,b
 integer                 :: t
  t = a
  a = b
  b = t
endsubroutine int_swap

subroutine real_swap (a,b)
 real, intent(in out) :: a,b
 real                 :: t
  t = a
  a = b
  b = t
endsubroutine real_swap

subroutine complex_swap (a,b)
 complex, intent(in out) :: a,b
 complex                 :: t
  t = a
  a = b
  b = t
endsubroutine complex_swap

subroutine logical_swap (a,b)
 logical, intent(in out) :: a,b
 logical                 :: t
  t = a
  a = b
  b = t
endsubroutine logical_swap

subroutine character_swap (a,b)
 character(len=*), intent(in out) :: a,b
 character(len=1)                 :: t
  if (len(a) == len(b) ) then
   do i = 1, len(a)
    t = a(i:i)
    a(i:i) = b(i:i)
    b(i:i) = t
   end do
  else
   print *, "Length of character strings differ."
   print *, "Can not swap."
   stop
  end if
endsubroutine character_swap

end module swap_module

program try_swap
 use swap_routines
 integer :: i1, i2
 character (len=16) :: c1, c2

  i1 = 10
  i2 = 20
  c1 = "string1"
  c2 = "the other string"

  print *, "i1 and i2 before swap", i1, i2
  call swap (i1, i2)
  print *, "i1 and i2 after swap", i1, i2

  print *, "c1 and c2 before swap>", c1,"< >", c2,"<"
  call swap (c1, c2)
  print *, "c1 and c2 after swap>", c1,"< >", c2,"<"

end program try_swap
```

F provides the ability to overload the intrinsic operators as long as the operators are not already part of F. For example, "+" can be overloaded to operate on two characters but it cannot be overloaded to operate on two integers because this would replace the intrinsic addition operation already found in F. F also provides the ability to create your own unary and binary operators as long as they are given names surrounded by dots as in ".inverse.".

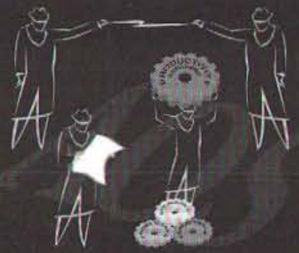
SmalltalkAgents®

Integrated Development Environment

for

Macintosh

Windows 95/NT



Check out what's new. . .

www.SmalltalkAgents.com

QKS

1405 Main Street
P.O. Box 371478
Montara, CA 94037-1478 USA

© Quasar Knowledge Systems, Inc. 1997
Windows 95 and Windows NT are Trademarks of Microsoft.

When overloading a name or an operator, all desired permutations of intrinsic and user defined types must be listed so that resolution to a specific function is always known at compile time; there are no run time resolutions in F. We can only guess that the Green team decided to do away with overloading operators out of the frustration created by C++ when a programmer attempts to figure out what A()+B() actually means. With its virtual functions, the expression A()+B() in C++ could be anything from a simple addition to a call to practically any function. It may require a thorough understanding of the design and most of the code to be sure exactly what A()+B() means. In F, one can conceivably click the "+" and be directed to the specific function that is being referenced.

Kinds Versus Specific Arithmetic

Java solves the non-portability created by different machine's mathematical models by requiring a specified range and precision for each type; if machines do not match this mathematical model, they must emulate it. F solves this same problem by allowing specifications of different kinds for types and literals. A kind number can be determined using the intrinsic procedures. For example, the expression

```
selected_real_kind(10,50)
```

returns a kind number to specify variables to have at least 10 digits of precision and an exponent range of at least 50. The kind number is then used in the declaration. For example:

```
module kind_module
 integer, public, parameter :: &
   k10_50 = selected_real_kind(10,50)
endmodule kind_module

module m_do_something_with_my_float
 use kind_module
 private ! do not export entities from kind_module
 real (kind = k10_50), public :: my_float
 ! ...
endmodule m_do_something_with_my_float
```

End-Of-Line Versus Semicolons

Unlike C, C++ and Java, statements in F conclude at the end-of-line unless an ampersand (&) is placed at the end-of-line to signal that this statement continues on the next line. Considering that most statements fit on one line, we feel that this design is easier for students to learn and less error-prone for both beginners and professionals. This may be a small point as today's C, C++ and Java programmers have obviously learned to cope with semicolons and since they are required inside certain statements (like the for statement) it appears that the Java designers had no choice.

Emphasis On Safety and Readability

The final difference between F and Java that we would like to mention is the overall strategy of safety and readability that was carefully designed into F. One way to compare this with the overall strategy of the "look-and-feel like C++" carefully designed into Java is that the F design is less marketable to today's programmers. Unfortunately for the software crisis, there are plenty more C++ programmers in the world than there are programmers that want to follow particular style guidelines.

For example, programmers with a preference for a one-line if statement do not care for the idea in F of requiring an endif statement. The required if-endif pair may lead to clearer blocks of code and unambiguous else statements, but more importantly it exemplifies the design strategy in F that abbrev.R!++ (abbreviations are not good). Requiring an endif statement in F sometimes means splitting one longer statement into three shorter statements by adding two words "then" and "endif". These words are important for error detection as well as tool writing. A missing parenthesis is easier to report if the reserved word "then" is required and a path coverage tool is easier to write if the word "endif" is required.

Our claim is that the safety and readability required of F code aid students and professionals. Possibly the best example of the safety designed into F is the intrinsic statement. Intrinsic procedures in F are allowed to be overloaded as long as the argument types differ from those found as part of the F language definition. For example, cos can be overloaded to accept a character argument but it cannot be overloaded to accept a real argument. Since there are so many intrinsic procedures, we decided that overloading one of them requires listing the name of the intrinsic on an intrinsic statement. This prevents the unknowing programmer from creating an involved module using a name that is already part of the language. Without this requirement, it would have been possible to overload cos, presume a reference to it, accidentally use the wrong argument and wind up referencing the intrinsic without ever realizing it.

Attention Teachers and Beginners

Data Encapsulation

It seems that right now everybody on the planet is interested in learning Java. Java has become so popular that many colleges and high schools are dropping Pascal and throwing beginners directly into Java. Though we believe Java is much more structured and friendly than C++, the fact remains that beginners need nurturing error messages more than the semester's survivors need the promise of a potential summer job. Indeed, good teachers can teach anything and good students can learn anything, but F offers a chance for those in the middle of the bell curve to learn how to program.

F jumps in between the academic emphasis of Pascal and the professional look and feel of Java to offer a compromise designed for both camps and ideally suited for potential engineers and scientists. The most attractive feature of object-oriented programming for beginning programming courses is data encapsulation. With its module-oriented programming, F is ideally suited for teaching and learning data encapsulation without getting lost in complicated inheritance chains.

Nurturing Versus Sink-or-Swim

Keep in mind that pure beginners have dozens of skills to learn, including:

- an editor (as well as using a keyboard),
- an operating system or environment,
- language syntax,
- deciphering cryptic compiler (or linker) error messages,
- **save, compile** and **link** commands or **save** and **run** commands,
- errors in logic, and last but not least
- programming concepts.

Minimizing the impact of this list was built into the design of F. On top of the F programming language sits a tutor/environment called F_world. F_world guides the pure beginner through writing and running their first F program. This program adds numbers until a zero is entered, uses a module with a subroutine, requires a loop and an if block and introduces one half of the F statements. Although this sounds complicated, it consistently creates programmers ranging from age eight to eighty in less than an hour. Listing 4 shows the statements that are fed one at a time to the beginner and a possible solution if the suggested names are copied.

Listing 4: F_world.f

F_worldBeginner

The F_world tutor's beginning program template and the suggested solution.

```
program ____________
 use ____________
  call ________________
endprogram ____________

! This module is entered into its own window
module ____________
 public :: ________________
```

```
contains
 subroutine ________________
  integer :: __________
  integer :: ___________
  _____________
  do
    print _____________________________
    read ____________
    if ________________ then
     exit
    else
     ____________________________
     print ______________________________
    endif
  enddo
  print ______________
 endsubroutine _____________
endmodule __________

! If the example names are used, here is the resulting F code
program MyFirstProgram
 use MyFirstModule
 call MyFirstSubroutine()
endprogram MyFirstProgram

! This module is entered into its own window
module MyFirstModule
 public :: MyFirstSubroutine
contains
 subroutine MyFirstSubroutine()
  integer :: input_number
  integer :: current_total
  current_total = 0
  do
    print *, "Enter number to add, or 0 to exit:"
    read *, input_number
    if (input_number == 0) then
     exit
    else
     current_total = current_total + input_number
     print *, "the current total is: ", current_total
    endif
  enddo
  print *, "Goodbye for now."
 endsubroutine MyFirstSubroutine
endmodule MyFirstModule
```

The F_world tutor uses the opposite strategy of "Hello World" and begins with a nontrivial example that can be used as a base for learning the basic concepts of programming in the first week if not in the first few hours.

The words **save, file, compile** and **link** are not used in F_world. Modules are entered into "windows" and **validated** and programs are simply **run**. As with importing packages in Java, using modules in F is a compile time feature; F_world knows where to find the necessary modules and "link" them with the program so the programmer does not have to. In F, all procedure interfaces are checked at compile time; there is no such thing as a linker error.

Natural branches from the tutor's beginning program include a description of the statements, loops, if-else decisions, programs, modules and subroutines, variables (we call them "containers"), integer data type, input and output. Playing with the code easily introduces other data types such as reals, subtraction instead of addition, functions instead of subroutines, private versus public and passing arguments to and from procedures to name a few. This normally covers the bulk of a semester when following the agenda set for programming courses two decades ago; with F_world we propose to cut this time in half, using the saved energy for more learning material or for learning Java!

INTERNET ™
SOURCE BOOK
The Executive's Guide to the Internet/Intranet World

3 COMPLETE GUIDEBOOKS IN ONE!
The First and Only Business-to-Business Directory for the Internet/Intranet World!
Over 6,000 company listings. 440+ categories
The Most Complete Online Shopping Guide
The WebShopping guide listing 7,000+ sites. 200+ categories
Find the Top 2,000 Corporate Websites!

MULTIMEDIA ™
S O U R C E   B O O K

If Has Anything to Do With Multimedia Chances Are You'll Find It Here!
The Best Business-to-Business Directory for the Multimedia World!
Over 6,000 company listings. 380+ categories
Access the World's Top Quality Suppliers of Services and Equipment In Multimedia Today!

The Place to Find It! The Place to Be Found!

Hi-Tech Media, Inc. 445 Fifth Ave., 27 Fl., NYC, NY 10016
Tel: 212-447-6400 800-663-5421 Fax: 212-447-9177
E-Mail: info@hi-techmedia.com www.hi-techmedia.com

Error Messages

Another key design goal of F was to be able to give specific error messages in as many situations as possible. The result is a grammar with minimized token look ahead requirements and an emphasis on keywords instead of punctuation and abbreviations. Since most statements begin with a reserved word, which is followed by specific tokens, likely errors such as misspelled keywords can be clearly reported. For example:

```
  intger :: total
! ^   ^
!ERROR (#41355) on line number: 5 !!!!!!!!!!!!!!!!!
!The word INTEGER is misspelled.            !!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Although sometimes trickier to diagnose, missing commas, colons and parentheses are for the most part also clearly reported. Since compiler writers most often have their hands full with simply getting the answers right, products for complicated programming languages often have few resources allocated to consistent and clear error reporting and recovery. Indeed, writing and reviewing error messages is often the last item "thrown in" before product testing. For example, one author (DE) misspelled the word intger in his first program and required 24 hours and help from another student to discover what "Bad statement syntax" meant when it was placed at the beginning of the one-erroneous-character Fortran 77 program.

Functions Are Not Subroutines

In F, procedures are either functions or subroutines. Functions are not permitted to have side effects so all arguments must be specified to have intent in. Subroutine arguments must be specified to have intent in, out, or inout. The attention given to argument intention and differentiating between functions and subroutines will surely help a student to understand what the word "void" means when placed in front of a Java function. In contrast, students beginning with Java will most likely think of void as required syntax until they are forced to learn its meaning.

From Supersets to Subset and Near Subset

Another View of F

It is not a coincidence that a language originally designed for non-computer scientists has benefits for both beginners and professionals. Fortran's history lends over forty years of language evolution targeting engineers/scientists/mathematicians/physicists/chemists/etc. The result is a language that contains a simplified syntax designed for problem solvers who do not have the desire to learn excruciating details and pitfalls of the implementation language. Even with its target market of non-programmers, Fortran 95 can be considered overly complicated since it contains all of Fortran 77 along with the recent additions from the '80s and '90s.

The mixture of the outdated 1977 features with the modern 1995 features essentially makes for two complete languages; the now outdated language was even considered lacking when introduced two decades ago. By carefully selecting the simplest subset of Fortran 95 without removing any desirable features, F presents an efficient package for people not wanting to carry around the old baggage of Fortran 77. An F processor will not accept a Fortran 77 program.

Another View of Java

A similar story can be told for Java. The creator of C++ was not motivated (and actually found it inappropriate) to remove the undesirable baggage C had been carrying around since the '70s. The creator of Java essentially had two complete languages to chisel into a smooth subset before realizing that sculpting a new masterpiece better fit his vision. Java is the first language in the evolution of C that emphasizes simplicity. A Java processor will not accept a C program.

Attention Professionals

You have seen how a simplified grammar aids the beginner but may be wondering what are the advantages for the professional. If you have ever picked up tens or hundreds of thousands of lines of code that were written by "somebody else" you may recall the pain related in simple tasks like finding the declaration of a variable or procedure. As business dictates, popular programming languages feed tool vendors with marketing ideas for reducing the pain of reading, updating and validating somebody else's code (if not your own). These tools, however, cost money and learning curve energy (often just to prove that the features you desired most will not be available until the next release of the tool).

Finding Something

While writing the F compiler front-end in F, we came across the need to find all array declarations. It did not take long to realize that a search for the word "dimension" was all we needed. The number of occurrences of the word "dimension" in quoted strings and comments were rare, but this potential problem when using a simple editor search is easily avoided by any serious programming team willing to spend a few days (or less) to write their own F scanner.

Large Projects

The idea of writing your own language tools designed for your specific project is powerful enough that we expect free F scanners written in F to become available soon. Given a tool that creates an array of tokens, finding all array declarations and other queries can be done with exactness. This exactness is required for accuracy of more involved homegrown projects such as profilers, static analyzers, test case coverage and test case creation tools.

Even without writing specific software tools, an editor is enough for large project programmers to benefit from the exactness required in F. For example, a debugging session may lead to a screen of code that calls subroutines Hidden, FindMe and WhereAmI. The first occurrence of these names in this module will tell if they are local and, if they are local, whether they are public or private.

It is quite convenient to discover that Hidden is private to the current module because all references will be found without having to open another window or file. This shows how the public and private differentiation aids with maintenance and code surfing as well as with design and encapsulation. Neither Fortran 77 nor C had the public and private differentiation.

If FindMe is not local, a search for the string "subroutine FindMe" will likely find the declaration. As well, while looking at the code for FindMe, searching for the same string will likely find the endsubroutine statement. This is, of course, the benefit of using the word "endsubroutine" followed by the subroutine name instead of the character "}" or the word "end".

Finally, if finding "subroutine WhereAmI" does not lead to the procedure code, it will lead to the interface that declares this procedure to be code that is written in a programming language other than F such as Fortran, High Performance Fortran (HPF), C or Java. The ability to call non-F procedures is only available in the professional edition of the F compilers.

Portability and Standards

The Fortran standard is constantly being updated with new features. As mentioned, this strictly enlarging nature of the Fortran standard can lead to an overly large language. Considering the alternative, undisciplined approach of creating languages and products before reaching a standardized consensus, the standardization process is one of Fortran's strengths.

Fortran vendors are relying on the standards teams and announcing new compilers only after the specifications have been accepted. Java's forced mathematical module and the bytecode-JVM provides portability; Fortran's standard's work and F's commitment to extract the safest features without compromising power provides both portability and efficiency. Before Fortran

2000, another push for portability is being made with the addition of Part 3 of the standard regarding conditional compilation.

The Future

Some may feel that a technology that comes as quickly as Java is bound to disappear just as quickly. The authors see, however, that Java is an improvement to C++ and is bound to persist and evolve for decades. With the Fortran committee's current focus on more powerful object-oriented features in Fortran 2000, F is also positioned to persist and evolve for decades.

Whether or not F can actually figure into the future of Java is yet to be seen. Java is currently lacking the efficiency that many large projects demand and the addition of JavaScript to the Java technology shows how quickly the Java folk are plugging in any existing holes. With four decades of compiler optimization backing Fortran, F surely presents the efficiency that Java lacks. Paradoxically, the issue of efficiency usually involves the misconception that Java lacks pointers, but the dereferencing mechanism found in Java is not all that different than the pointer-target mechanism found in F.

Summary

It is difficult to summarize this presumably shocking report that Java is not all that different than Fortran and its subset F. There are far too many similarities and differences between F and Java to give a full detailed description in this article. We expect that this discussion has raised a few eyebrows.

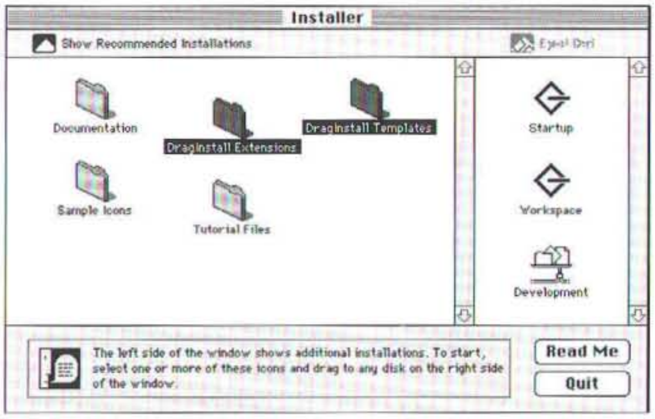
Scientists and engineers will no doubt find F to be a similar look and feel to Fortran 77 as computer scientists find when moving from C or C++ to Java. Fortran 77 programmers have some learning to do before writing F code and C programmers have more learning to do than C++ programmers before writing Java code. F is even more familiar to those rare Fortran 90/95 programmers since every F program is also a Fortran 90/95 program.

A fitting ending to this F introduction and comparison to Java may be the promise that the next generation of programmers, whether computer scientists or others, are learning well structured module-oriented and object-oriented techniques from the very start. This is accomplished by beginning with F and moving to Java, all in their first year of programming. As this scenario unfolds in the academic world, the average student and not just the "smart kids" can be given a chance to learn what programming is all about. After all, programming really is not all that difficult and it is finally becoming simpler.

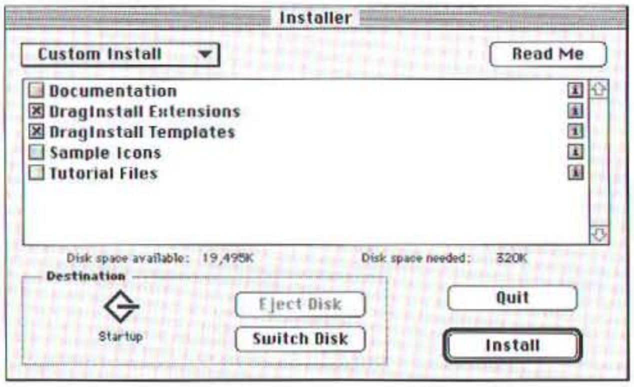
Acknoledgements

Thanks to Absoft Corp. for helping to make F available on the PowerPC Macintosh. Thanks also to Numerical Algorithms Group, Inc. (NAG), Fujitsu Limited, and Salford Software, Inc. for working with Imagine1 to make F available on those other systems (Windows, Linux and Unix.) Mike Dedina helped review and format this report. Thanks goes out as well to the Fortran community for providing immediate interest in the F programming language. MT

Guess which installer was built by DragInstall®



The left side of the window shows additional installations. To start, select one or more of these icons and drag to any disk on the right side of the window.



☐ Documentation
☒ DragInstall Extensions
☒ DragInstall Templates
☐ Sample Icons
☐ Tutorial Files

Disk space available: 19,495K Disk space needed: 320K

Guess again.

If you guessed that the first installer was built by DragInstall, you're only half right. Now with DragInstall 3.0, you can build *both* of these installers from a *single* script file!

For more information about DragInstall and a free demo, visit our web site at **http://www.sauers.com/draginstall** or give us a call at **1-800-890-9880**.

by Andrew Downs

Calling C Code from Java

Using native code with Java applications and the Sun JDK

Introduction

This article shows how to access C code (also called native code) from within Java code. The C code exists as a PowerPC shared library containing several Process Manager calls. The functions in the shared library are used to get and store process information. A Java front-end calls the shared library at specified intervals to retrieve and display the process information. Timing for these calls is accomplished using a thread. The front-end also includes a preferences dialog with a pseudo-custom control written in Java. This program runs as a Java application, not an applet.

This article assumes you are familiar with both Java and the Macintosh Toolbox. The Java portion of this program was developed using the Sun Java Development Kit (JDK) version 1.0.2. The native code is written in C. CodeWarrior 9 is used to build a PowerPC shared library from the C code.

Why Native Code?

One of Java's strengths is that it often insulates the programmer from the specifics of the users' platform. Sometimes, however, you need access to information that cannot be retrieved using one of the existing Java packages.

Fortunately, Java allows you to call non-Java code, usually referred to as native code. Using the Sun JDK v1.0.2 on the Macintosh, this native code must exist as a PowerPC shared library. If you have access to one of the non-Java integrated development environments, creating such a library is available as a project option. Refer to your development environment information for details. This article will use CodeWarrior to create a shared library. A sample CodeWarrior project for this purpose is also included with the JDK.

Briefly, here are the steps in the overall development process:

1. Write and compile the Java code that makes the call to a native method. If the file is called <native>.java, the generated code will be contained in <native>.class.
2. Drop the <native>class file onto the JavaH application (included with the JDK), which will generate the header and stub files.
3. Move the <native>.h and <native>.stub files into the folder with your native code.
4. Build the native library in CodeWarrior, using <native>.c as the source file.
5. Move the native library (or an alias to it) to one of two places: the JavaSoft Folder in the Extensions folder, or the folder containing the Java front-end. (These locations are mapped into the classpath by Java Runner, so it can find any supporting code.)
6. Write and compile the Java code for the front-end.
7. Drop the .class file containing your application's main() method onto the Sun Java Runner application to run it.

Before we begin, let's take a closer look at shared libraries and what should be put in them.

What's in a Shared Library?

A shared library consists of functions that may be called by one or more applications (or other types of code, including other libraries). The shared library functions do not run as standalone code. Rather, the shared library provides its services to applications that know how to call its member functions. Shared library code exists in a separate file from the code fragment or fragments that comprise the application.

Andrew Downs <andrew.downs@tulane.edu> is a programmer for the Office of Medical Informatics at the Tulane University School of Medicine in New Orleans, LA. He also teaches C and Java programming at Tulane University College. Andrew wrote the Macintosh shareware program Net Manager, and the Java application UDPing.

When creating a PowerPC shared library, the function code gets compiled into the data fork of the library file. The data fork acts as a container for the compiled code. (Multiple containers may exist in the data fork, but we won't deal with that issue here.) A code fragment resource (of type 'cfrg') with an ID of 0 (zero) is placed in the resource fork of the library file. This resource contains information about where in the data fork the library code begins, as well as what architecture the code adheres to (in this case, PowerPC).

The shared library must be loaded into memory before use. This may be done when the client application starts up, or as requested by the application. Our Java code relies on the latter approach: one of the classes contains a call to load and prepare the library at a specific time during program execution. The Process Manager (which handles the launching and running of applications) relies on the services of the Code Fragment Manager to prepare the library code for execution. (For more information on the Code Fragment Manager, refer to *Inside Macintosh: PowerPC System Software.*)

You will see that the source code for our shared library does not resemble a typical Macintosh application. It has no event loop, and does not initialize any of the Toolbox managers. Instead, the functions in our shared library rely on the client application to set up any managers that may be required.

The current version of the JDK does not contain any packages or classes that let you peek at the currently running processes on the system. That will be the purpose of our shared library: to obtain and provide process information to our Java front-end program.

Getting the process information into the shared library is easy. A function in the shared library calls the appropriate Process Manager routines, and saves the data they return.

We need a way to get this process information from the shared library to our Java program. We accomplish this using a Java class containing definitions of native methods. (The word "native" is a modifier in Java which may be used in method declarations. It indicates that the body of the method is defined elsewhere, and is not Java code.) These native methods are really calls to C functions in the shared library. We will treat them like accessor methods: each is responsible for one action, such as returning to the caller a specific piece of data. One of them instructs the shared library to read a new set of process information. Another returns the number of current processes. A third returns the partition size of a specific process, and so on.

Setting up the Shared Library

At this point, we know that we have to create a shared library containing C code, and that the functions in this library will be called using native methods in a Java class. We determined that the functions will return specific pieces of data. We can now design the shared library in more detail.

The Process Manager information we need includes the name, partition size, and current RAM in use by each process. Once we get the information, we store it in a (global) array of structs. Each struct contains the data for one process. To get at the data, we will walk through the array, checking each element in turn.

We can setup the header file for our C code as follows:

Listing 1: ProcessWatcher.h

ProcessWatcher.h

The header file for ProcessWatcher.c.

```
/**************************
  ProcessWatcher.h
**************************/
// Define some macros to make our code more readable.
#define  NIL                  0L
#define  ONE_KBYTE_SIZE       1024
#define  DIFFERENCE_FACTOR    16
#define  MAX_NUM_RECORDS      20
#define  STRING_LENGTH        32

// Define a struct that holds the information we want for
// one process.Then create an array of these structs.
struct appStruct
{
  Str32  processName;
  long   processSize;
  long   currentSize;
} ;

struct   appStruct  gArray[ MAX_NUM_RECORDS ];

// One global variable as an array counter.
short    gNumRecords;

// Our function prototypes.
long   DoGetNumRecords( void );
long   DoGetCurrentSize( long i );
long   DoGetProcessSize( long i );
long   DoGetProcessNameLength( long i );
long   DoGetProcessNameChar( long i, long j );
void   DoGetProcessInfo( void );
```

The first five function prototypes listed above retrieve array data for the caller (in our program, the Java front-end). The last prototype corresponds to the function which actually calls the Process Manager.

Notice that all of the return values are long integers. This is because of the translation of Java primitive data types to Macintosh data types, which we will examine shortly.

Now we can look at the implementation of those functions. This is a partial listing of the contents of ProcessWatcher.c. We will view the remainder shortly.

Listing 2: ProcessWatcher.c

ProcessWatcher.c

Retrieve and save information about currently running processes.

```
/**************************
  DoGetNumRecords
**************************/
long  DoGetNumRecords( void )
{
  return( ( long )gNumRecords );
}

/**************************
  DoGetCurrentSize
**************************/
long  DoGetCurrentSize( long i )
{
  return( gArray[ i ].currentSize );
}
```

Visit MacTech® Magazine's Web site!
http://www.mactech.com

```
/**************************
  DoGetProcessSize
**************************/
long   DoGetProcessSize( long i )
{
  return( gArray[ i ].processSize );
}

/**************************
  DoGetProcessNameLength
**************************/
long   DoGetProcessNameLength( long i )
{
  return( gArray[ i ].processName[ 0 ] );
}

/**************************
  DoGetProcessNameChar
**************************/
long   DoGetProcessNameChar( long i, long j )
{
  return( gArray[ i ].processName[ j ] );
}

/**************************
  DoGetProcessInfo
**************************/
// This routine walks through the current process list
// and retrieves information we need.
void   DoGetProcessInfo( void )
{
  FSSpec               theSpec;
  OSErr                theError;
  ProcessInfoRec       theInfoRec;
  ProcessSerialNumber  thePSN;
  Str32                theName;

  // Reset the array count.
  gNumRecords = 0;

  // Setup the record for retrieving process info.
  theInfoRec.processInfoLength = sizeof( ProcessInfoRec );
  theInfoRec.processName = theName;
  theInfoRec.processAppSpec = &theSpec;

  thePSN.highLongOfPSN = NIL;
  thePSN.lowLongOfPSN = kNoProcess;

  // Retrieve process info until no more processes are found.
  while ( GetNextProcess( &thePSN ) == noErr )
  {
    theError = GetProcessInformation( &thePSN, &theInfoRec );

    if ( theError != noErr )
      break;

    // Save the process data into the global array. Start with the process name.
    BlockMove( ( Ptr )theInfoRec.processName,
      ( Ptr )gArray[ gNumRecords ].processName,
      STRING_LENGTH );

  // Partition sizes are 16 bytes off when compared to "About This Macintosh".
  // Current sizes are 15 bytes off when compared to "About This Macintosh".
  // Adjust and store the resulting values so our display matches the system display.
    gArray[ gNumRecords ].processSize =
      ( theInfoRec.processSize / ONE_KBYTE_SIZE )
      - DIFFERENCE_FACTOR;
    gArray[ gNumRecords ].currentSize = (
      ( theInfoRec.processSize - theInfoRec.processFreeMem )
      / ONE_KBYTE_SIZE ) - DIFFERENCE_FACTOR + 1;

    gNumRecords++;

    // Bounds checking.
    if ( gNumRecords > MAX_NUM_RECORDS )
      break;
  }
}
```

This is a good start, so now we turn our attention to the Java class that contains the native methods. Remember, these are the Java calls to our C code.

We will abstract this portion of the design slightly to allow some flexibility. This means that the names of the native methods will not match the function names just described, although they have a similar purpose. Why is this? To allow us to make minor modifications to either piece without affecting the other. For instance, we can change the variable names in the C code, and the Java class still performs the same as before.

Let's take a closer look at the Java class containing the native methods.

NATIVE METHODS IN A JAVA CLASS

To use the shared library, Java requires a class definition that both loads the library and calls the native methods within it. It also requires method declarations inserted in the calling class file matching the prototype for the called function. Notice that these declarations contain arguments but no body, ending in a semicolon (similar to a function prototype). The keyword 'native' tells the compiler that the code for these methods will be defined elsewhere.

Listing 3 shows the Java code for our native class (NProcessWatcher). The purpose of the static initializer is to ensure that the shared library gets loaded when this class gets loaded. (For details on the loading process, refer to *Inside Macintosh: PowerPC System Software,* and its discussion of the Code Fragment Manager.)

Listing 3: NProcessWatcher.java

NProcessWatcher.java

The class that will interface to our shared library.

```
public class NProcessWatcher
{
  // Tell the VM to load our shared library.
  static
  {
    System.loadLibrary("ProcessWatcher Library");
  }

  // Here are definitions for several simple native methods. Note that the data types
  // we specify here may not be the same in the corresponding header file, due to
  // size differences between Java's primitive types and the Macintosh's data types.
  // The stubs file should handle thetranslation of the arguments and return types
  // for us.

  // This method invokes the Process Manager portion of the shared library.
  private native void nativeGetProcessInfo();

  // The following methods retrieve various pieces of information
  // about the current processes.
  private native int nativeGetNumElements();

  private native int nativeGetCurrentSize( int i );

  private native int nativeGetProcessSize( int i );

  private native int nativeGetProcessNameLength( int i );

  private native char
    nativeGetProcessNameChar( int i, int j );

  // Our class constructor. It calls our native method
  // that updates the Process Manager information.
  public NProcessWatcher()
  {
    this.nativeGetProcessInfo();
  }
  // Update the Process Manager information on request.
  public void updateNow()
  {
    this.nativeGetProcessInfo();
  }

  // Accessor methods used by our other Java classes to get process info. By hiding
  // the actual calls to the native methods, we can change the implementation
  // without affecting the classes calling in.
  public int getNumElements()
  {
    return( this.nativeGetNumElements() );
  }

  public int getCurrentSize( int i )
  {
    return( this.nativeGetCurrentSize( i ) );
  }

  public int getProcessSize( int i )
  {
    return( this.nativeGetProcessSize( i ) );
  }

  public int getProcessNameLength( int i )
  {
    return( this.nativeGetProcessNameLength( i ) );
  }

  public char getProcessNameChar( int i, int j )
  {
    return( this.nativeGetProcessNameChar( i, j ) );
  }
}
```

With this class defined, we can now compile it. It does not rely on any other classes, so it will compile by itself using the Java Compiler in the JDK.

We still need a way to tie together this class and our shared library. Specifically, we need two additional files. One is a second header file, containing the prototypes for the Java calls into the C code. The other is a stubs file that can handle the translation of arguments and return values between our Java and C code. The JDK provides a tool that will handle this for us: JavaH.

The JavaH utility generates the header and stub files from the class file containing the native methods. After you compile NProcessWatcher.java, drop the resulting class file (NProcessWatcher.class) on the JavaH icon. This should create NProcessWatcher.h and NProcessWatcher.stubs in the same folder as JavaH.

Move these two files into the folder containing your CodeWarrior project before building the library. You should not modify these files. However, we must use of the information in the header file. If you open NProcessWatcher.h, you will see the native method declarations, rewritten as C function prototypes.

Listing 4: NProcessWatcher.h

NProcessWatcher.h

The JavaH generated header file for our native methods. This listing is reformatted to fit the magazine page.

```
#include <native.h>

/* header for class NProcessWatcher */
#ifndef _Included_NProcessWatcher
#define _Included_NProcessWatcher
#ifdef __cplusplus
extern "C" {
#endif

typedef struct ClassNProcessWatcher {
  char pad[1]; /* Padding for ANSI C */
} ClassNProcessWatcher;

HandleTo(NProcessWatcher);

extern void NProcessWatcher_nativeGetProcessInfo
                (struct HNProcessWatcher*);
extern long NProcessWatcher_nativeGetNumElements
                (struct HNProcessWatcher*);
extern long NProcessWatcher_nativeGetCurrentSize
                (struct HNProcessWatcher*, long);
extern long NProcessWatcher_nativeGetProcessSize
                (struct HNProcessWatcher*, long);
extern long NProcessWatcher_nativeGetProcessNameLength
                (struct HNProcessWatcher*, long);
extern /*unicode*/ long
                NProcessWatcher_nativeGetProcessNameChar
                (struct HNProcessWatcher*, long, long);
#ifdef __cplusplus
}
#endif

#endif /* _Included_NProcessWatcher */
```

Copy the six prototypes over to ProcessWatcher.c, and paste them near the top, after the #includes. Delete the word "extern" at the front of each one. Now we can fill in the bodies for these functions. Here is the revised version of ProcessWatcher.c:

Listing 2 revisited: ProcessWatcher.c

ProcessWatcher.c

Retrieve and save information about currently running processes.

```
// Implementation of a native shared library for MacOS.
// Adapted from the example provided with the JDK v1.0.2.

// OS Headers
#include <CodeFragments.h>

// JavaH generated Header
#include "NProcessWatcher.h"

// Our library header.
#include "ProcessWatcher.h"

// The first functions listed are the calls into our native library. They match the
// prototypes in the header file NProcessWatcher.h, which we included above.

// The Java data types (int and char) expected as return types by NProcessWatcher.java
// map to the long int data type on the Macintosh. We didn't arbitrarily select this; it
// was generated by JavaH as part of NProcessWatcher.h

// We have functions calling functions to get the process data.
// This allows us to change the underlying implementation if necessary.

void NProcessWatcher_nativeGetProcessInfo
                (struct HNProcessWatcher* self)
{
  DoGetProcessInfo();
}

long NProcessWatcher_nativeGetNumElements
                (struct HNProcessWatcher* self)
{
  return( DoGetNumRecords() );
}

long NProcessWatcher_nativeGetCurrentSize
                (struct HNProcessWatcher* self, long i)
{
  return( DoGetCurrentSize( i ) );
}
```

```
long NProcessWatcher_nativeGetProcessSize
               (struct HNProcessWatcher* self, long i)
{
  return( DoGetProcessSize( i ) );
}

long NProcessWatcher_nativeGetProcessNameLength
               (struct HNProcessWatcher*, long i)
{
  return( DoGetProcessNameLength( i ) );
}

long NProcessWatcher_nativeGetProcessNameChar
               (struct HNProcessWatcher*, long i, long j)
{
  return( DoGetProcessNameChar( i, j ) );
}

/*************************
  DoGetNumRecords
*************************/

long   DoGetNumRecords( void )
{
  return( ( long )gNumRecords );
}

/*************************
  DoGetCurrentSize
*************************/

long   DoGetCurrentSize( long i )
{
  return( gArray[ i ].currentSize );
}

/*************************
  DoGetProcessSize
*************************/

long   DoGetProcessSize( long i )
{
  return( gArray[ i ].processSize );
}

/*************************
  DoGetProcessNameLength
*************************/

long   DoGetProcessNameLength( long i )
{
  return( gArray[ i ].processName[ 0 ] );
}

/*************************
  DoGetProcessNameChar
*************************/

long   DoGetProcessNameChar( long i, long j )
{
  return( gArray[ i ].processName[ j ] );
}

/*************************
  DoGetProcessInfo
*************************/

// This routine walks through the current process list
// and retrieves information we need.
void   DoGetProcessInfo( void )
{
  FSSpec              theSpec;
  OSErr               theError;
  ProcessInfoRec      theInfoRec;
  ProcessSerialNumber thePSN;
  Str32               theName;

  // Reset the array count.
  gNumRecords = 0;

  // Setup the record for retrieving process info.
  theInfoRec.processInfoLength = sizeof( ProcessInfoRec );
  theInfoRec.processName = theName;
  theInfoRec.processAppSpec = &theSpec;

  thePSN.highLongOfPSN = NIL;
  thePSN.lowLongOfPSN = kNoProcess;

  // Retrieve process info until no more processes are found.
  while ( GetNextProcess( &thePSN ) == noErr )
  {
    theError = GetProcessInformation( &thePSN, &theInfoRec );

    if ( theError != noErr )
      break;

    // Save the process data into the global array. Start with the process name.
    BlockMove( ( Ptr )theInfoRec.processName,
      ( Ptr )gArray[ gNumRecords ].processName,
      STRING_LENGTH );

    // Partition sizes are 16 bytes off when compared to "About..."
    // Current sizes are 15 bytes off when compared to "About..."
    gArray[ gNumRecords ].processSize =
      ( theInfoRec.processSize / ONE_KBYTE_SIZE )
      - DIFFERENCE_FACTOR;
    gArray[ gNumRecords ].currentSize = (
      ( theInfoRec.processSize - theInfoRec.processFreeMem )
      / ONE_KBYTE_SIZE ) - DIFFERENCE_FACTOR + 1;

    gNumRecords++;

    // Bounds checking.
    if ( gNumRecords > MAX_NUM_RECORDS )
      break;
  }
}
```

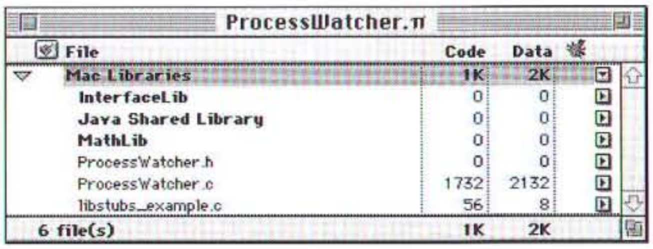
Each prototype contains the class name, followed by an underscore, then the name of the native method from the class. The parameters include a reference to the calling class, and any arguments. The first function is declared as void, and simply calls another void function in this c file. The other functions accept parameters and return values. We specified Java integers in the original method declarations, and we see here that they were adjusted (by JavaH) to Macintosh long ints in the header file to correspond to the 32-bit data type.

These functions call other functions in this file to get process data. Those values are then returned to the caller (NProcessWatcher).

One more step is required before we attempt to make the shared library project. Make an alias to the file "Java Shared Library", which is located in the path System Folder:Extensions:JavaSoft Folder. Place the alias into the folder containing the shared library project.

The Project Definition

Figure 1 shows the CodeWarrior project definition.



ProcessWatcher.π

File	Code	Data
Mac Libraries	1K	2K
InterfaceLib	0	0
Java Shared Library	0	0
MathLib	0	0
ProcessWatcher.h	0	0
ProcessWatcher.c	1732	2132
libstubs_example.c	56	8
6 file(s)	1K	2K

***Figure 1.** The CodeWarrior project.*

Here are several of the project preferences settings necessary to create the shared library:

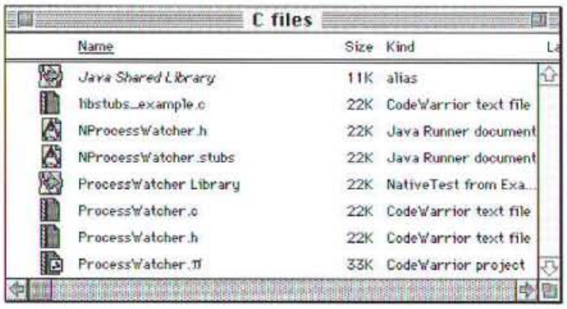
PPC Project:	
File Name:	ProcessWatcher Library
Creator:	Java
Type:	shlb
PPC PEF:	
Fragment Name:	ProcessWatcher Library
Library Folder ID:	0

Our shared library is actually a code fragment. The Preferred Executable Format (PEF) dictates the layout of the generated fragment. The Fragment Name will be used by the Code Fragment Manager to identify and load the fragment. We don't use the Library Folder ID setting: it is for specifying the ID of an alias resource, which points to a directory containing other libraries that our fragment uses.

We also do not use the version numbers (also in the PPC PEF window), preferring to leave them at 0 (zero). If you create additional versions of the library, and do not maintain backward compatibility, you can use version numbers to specify which versions of other fragments may call in to (or be called by) this fragment.

There are no initialization or termination entry points defined for this project. As indicated in the CodeWarrior documentation, such functions could be used to setup or teardown the resources or memory needed by the fragment. There is no main entry point for a shared library, and so we do not specify one.

Figure 2 shows the files used in creating the native library. (A copy of the shared library is also shown.)



C files

Name	Size	Kind
Java Shared Library	11K	alias
libstubs_example.c	22K	CodeWarrior text file
NProcessWatcher.h	22K	Java Runner document
NProcessWatcher.stubs	22K	Java Runner document
ProcessWatcher Library	22K	NativeTest from Exa...
ProcessWatcher.c	22K	CodeWarrior text file
ProcessWatcher.h	22K	CodeWarrior text file
ProcessWatcher.π	33K	CodeWarrior project

Figure 2. *The files used to build the native library.*

These are the contents of the files shown in Figure 2:

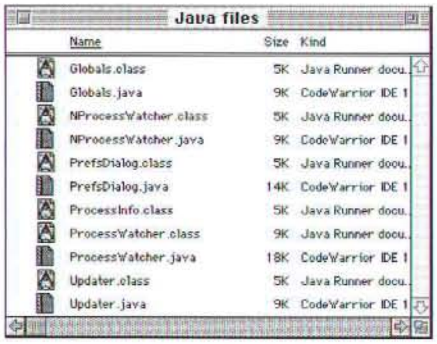
- Java Shared Library: an alias to a file in the JavaSoft folder, required during the build.
- NProcessWatcher.h: header for ProcessWatcher. Generated by JavaH.
- NProcessWatcher.stubs: stubs for ProcessWatcher. Generated by JavaH.
- ProcessWatcher.c: the calls to the MacOS ProcessManager.
- ProcessWatcher.h: constant definitions and function prototypes for ProcessWatcher.c.
- ProcessWatcher.π: project definition file.
- libstubs_example.c: export information for use by CodeWarrior.

For this last file, I kept the same name as the original included with the JDK, but modified the contents to include NProcessWatcher.stubs.

You can now make this project by pressing Command-M, which will compile and link the code. The compiler should place the shared library file "ProcessWatcher Library" in the project directory.

JAVA CLASS INTERACTIONS

We can now turn our attention to the Java portion of our program. We have already looked at NProcessWatcher, which contains the native method calls. Figure 3 shows the names of all of the Java class and source code files.



Java files

Name	Size	Kind
Globals.class	5K	Java Runner docu...
Globals.java	9K	CodeWarrior IDE 1
NProcessWatcher.class	5K	Java Runner docu...
NProcessWatcher.java	9K	CodeWarrior IDE 1
PrefsDialog.class	5K	Java Runner docu...
PrefsDialog.java	14K	CodeWarrior IDE 1
ProcessInfo.class	5K	Java Runner docu...
ProcessWatcher.class	9K	Java Runner docu...
ProcessWatcher.java	18K	CodeWarrior IDE 1
Updater.class	5K	Java Runner docu...
Updater.java	9K	CodeWarrior IDE 1

Figure 3. *The Java source and class files for the front-end.*

HIGH PERFORMANCE IMAGING AND ANNOTATION DEVELOPMENT TOOLS

RasterMaster 6.0 - Familiar API and well-proven technology in use world-wide by companies such as British Petroleum, Chase Manhattan, Eastman Kodak, Filenet, Ford, Gannett, Hewlett-Packard, IMSI, Lexis, Philips, Polaroid, Siemens-Nixdorf, Trix, Unisys, and Xerox.

50+ Raster Formats: All TIFF, JPEG, Group3, Group4, MO:DCA IOCA, CALS, PNG, PCX, BMP, GIF, PhotoCD, ABIC, **Flashpix** and more!

RasterNote™annotation/redlining toolkit - includes all popular features. Create Sticky Notes, Freehand Drawing, Lines, Ellipses, Polygons.

NEW ADDITIONS: RasterNote, UNIX, Alpha, Flashpix
Format Reading, Saving, Compression, Display, Despeckle
Rotate, Zoom, Pan, Scroll, Deskew, Anti aliased display,
Auto aspect ratio, Image processing, Transparent color
Improved Twain scanning, Color reduction and promotion

Platforms: Win 95, Win NT, Win 3.1, OS/2, Alpha, Macintosh, UNIX
Environments: DLL, VBX, OCX/ActiveX, Delphi, VB, PBuilder, VC ++

Call 617 630-9495 or see our Website for FREE, NO RISK EVALUATIONS!
www.snowbnd.com

PO Box 520 Newton, MA 02159 Tel: 617 630-9495 Fax: 617 630-0210
Email: salesmc@snowbnd.com

SNOWBOUND SOFTWARE...
Imaging Software - Powerful, Fast, Reliable

Here is the purpose of each Java source code file:

- Globals: constants used throughout the other classes.
- NProcessWatcher: calls the native code shared library.
- PrefsDialog: the preferences dialog.
- ProcessInfo: data storage for information about each process.
- ProcessWatcher: the front-end and main window.
- Updater: the thread which periodically calls NProcessWatcher.

Figure 4 illustrates the interaction between the Java classes and the library.

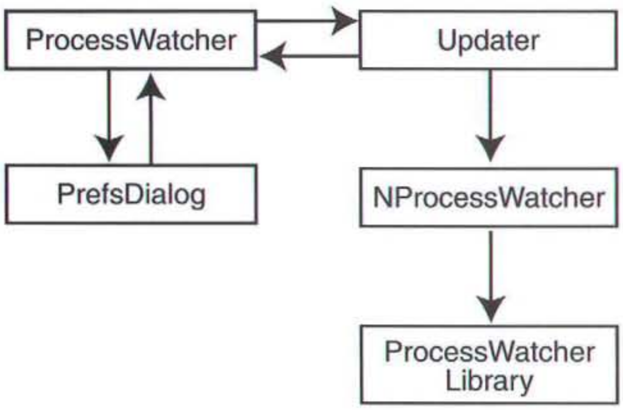


Figure 4. *Class file and library interaction.*

These interactions are defined as follows:

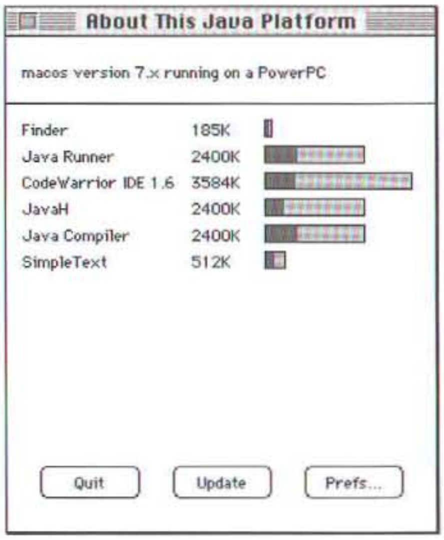
1. The Java front-end(ProcessWatcher) creates a Java thread (Updater), and starts it running.
2. Updater creates an instance of the class NProcessWatcher, which interfaces with the shared library. NProcessWatcher loads the PPC shared library (ProcessWatcher Library).
3. NProcessWatcher calls a native function within ProcessWatcher Library, instructing it to retrieve and save the current Process Manager information.
4. Updater tells ProcessWatcher to request and display the process information.
5. Updater goes to sleep. On wake up, return to step 3.

In addition, ProcessWatcher calls PrefsDialog when the user clicks the Prefs button. If the user updates the settings, the changes are signaled back to ProcessWatcher.

Want to know what products are available for MacOS development? Check out Developer Depot™ <http://www.devdepot.com>

The Java Front-End

The Java front-end is a window that looks vaguely similar to the "About This Macintosh" window under the Apple Menu (refer to ***Figure 5***).



macos version 7.x running on a PowerPC

Finder	185K
Java Runner	2400K
CodeWarrior IDE 1.6	3584K
JavaH	2400K
Java Compiler	2400K
SimpleText	512K

Figure 5. *The finished product.*

When launched, the front-end retrieves and displays some information about the operating system and architecture. Then it creates a thread that calls the native code in the shared library. The thread instructs the front-end to update its window, then goes to sleep for a user-specified number of seconds. When it receives the update message from the thread, the front-end communicates with the shared library using the native methods described previously. It retrieves a count of the number of processes, then the partition size, RAM in use, and name for each process. The front-end then draws RAM usage bars in its window.

The front-end has three buttons near the bottom: Quit, Update, and Prefs. Quit leaves the application. Update performs an immediate update of the data displayed in the window. This is the same action taken when the thread informs the window to update, but using the button it can be done on demand. Clicking the Prefs button displays the preferences dialog.

For simplicity, the user interface items are placed using absolute values (screen coordinates). Note that this may affect the look of the front-end if you use it on a non-Macintosh platform. (Of course, you would also have to rewrite the shared library for the new platform.)

The preferences dialog is a moveable modal dialog, as shown in Figure 6. It contains a text field, a scrollbar, and Cancel and OK buttons. The text field shows the sleep setting for the thread (in seconds). It is set to non-editable, and its contents are highlighted. It gets updated with new values as the user moves the scrollbar. The scrollbar is horizontal. Larger values are to the right. Clicking Cancel will dismiss the dialog without saving any changes the user made to the sleep setting. Clicking OK will save the changes, then dismiss the dialog. Settings are kept internally; they are not written out to a preferences file.

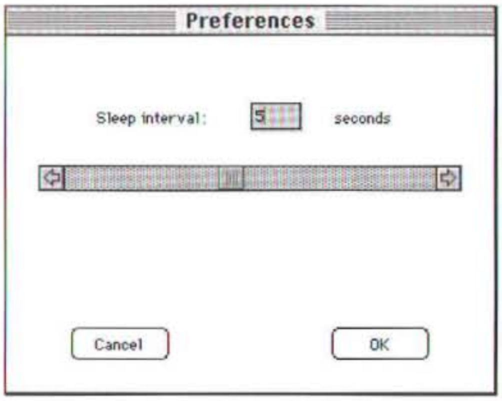


Figure 6. *The Preferences dialog.*

The scrollbar event-handling is related to the text field in this way: when the user clicks in the scrollbar or moves the thumb, the appropriate event is caught, and the number in the text field changes appropriately. Currently, page up and page down events are not handled. Scrollbar event handling is accomplished by overriding the handleEvent() method in PrefsDialog.

The updater thread leads a simple life. Once created, it goes through a continuous cycle. It first creates an instance of the native class, which is responsible for calling the native methods (which invoke the C functions). Next, the thread tells its parent (the front-end) to update its display. Finally, the thread goes to sleep for the user-specified sleep interval. When it wakes up, if the native object has been destroyed for some reason, it gets recreated. Otherwise, the thread instructs both the native object and the parent object to perform their respective update operations.

Listing 6 contains the code for requesting process information from the shared library, and the paint() method used to display it. Here is the method that requests the process information. Notice that it retrieves the process names one character at a time into a character array, then converts that array to a String.

Listing 6: ProcessWatcher.java (partial listing)

ProcessWatcher.java

The front-end for our Java application.

```
public void readProcessInfo()
{
   // Retrieve the reference to our native class, maintained by the Updater.
   NProcessWatcher tempNProcessWatcher =
     theUpdaterThread.doGetNativeClass();

   // If the native class has not been instantiated yet, leave.
   if ( tempNProcessWatcher == null )
     return;

   // Clear out the Vector.
   theProcessList.removeAllElements();

   // Counters for traversing the array of processes,
   // and the individual process names.
   int i = 0, j = 0;

   // So we know when we've retrieved all the process elements...
   int max = tempNProcessWatcher.getNumElements();

   // Walk through the process elements one at a time.
   for ( i = 0; i < max; i++ )
   {
      // Retrieve process size info from the native class.
      int theCurRAM =
            tempNProcessWatcher.getCurrentSize( i );
      int thePartition =
            tempNProcessWatcher.getProcessSize( i );

      // How long is the process name?
      int theLength =
            tempNProcessWatcher.getProcessNameLength( i );

      // We'll hardcode a max name size. It must be at least as
      // large as the name size in the C code (32).
      char[] theCharArray = new char[ 32 ];

      // Get the process name, one character at a time. Note the dual
      // counters used to get the char:
      //    i is the current process array element, and
      //    j + 1 is a character in the name (Str32) for that process.
      // Our character array starts its element numbering at 0, but the Str32 data
      // type (used in the C code) uses location 0 for the length of the string, so
      // we have to adjust the counter j to start one element beyond it.
      for ( j = 0; j < theLength; j++ )
        theCharArray[ j ] =
        tempNProcessWatcher.getProcessNameChar( i, j + 1 );

      // Once the char array is filled in, create a Java String from it.
      String theNameString = new String( theCharArray );

      // Create a new instance of the ProcessInfo class, fill in
      // the values, then place the reference to the object into the Vector.
      ProcessInfo tempProcessInfo = new ProcessInfo();
      tempProcessInfo.setCurRAM( theCurRAM );
      tempProcessInfo.setPartition( thePartition );
      tempProcessInfo.setName( theNameString );
      theProcessList.addElement( tempProcessInfo );

      // Force changes to redraw.
      repaint();
   }
```

Listings 6 through 8 contain the Java code for the front-end, preferences dialog, and updater thread, respectively. Listing 9 contains definitions for various constants used throughout the Java classes. These listings are fairly well structured, and should be easy to read.

Conclusion

Calling C code from Java requires some preparation. You must properly construct a PowerPC shared library and define methods within your Java classes for calling this library. We explored the steps necessary to get data from the Process Manager to a shared library and then to a Java front-end for display. In situations where you need Macintosh Toolbox functionality that Java does not inherently provide, a shared library can be an effective solution. This approach is supported by Sun, so it makes sense to use it if necessary.

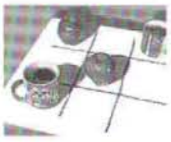
References

1. Inside CodeWarrior 8, Metrowerks Inc., 1996.
2. Inside Macintosh: PowerPC System Software, Apple Computer, Inc., Addison-Wesley Publishing Company, 1994.
3. Teach Yourself Java in 21 Days, Laura Lemay and Charles L. Perkins, Sams.net Publishing, 1996.

URLs

Source files: <http://www1.omi.tulane.edu/adowns/MacTech/source/>.

by Danny Swarzman, Stow Lake Software



Putting Java under PowerPlant

Building a strategic game application with a C++ engine and a Java user interface

PREFACE

With *Mac OS Runtime for Java™(MRJ)* (Pronounced 'marge') you can use C++ to develop a Macintosh application which runs Java code. The Java part could be an applet, an application or neither. MRJ is delivered as shared libraries. The program interface, JManager, is supplied in the MRJ SDK. Both are available from Apple's Java website <http://appleJava.apple.com/>.

With MRJ you create a custom Java runner. It could be general-purpose or, as described here, designed to run a particular Java program. There are many reasons why you may want to do this. For example, you may have some legacy code in C or C++, such as an engine which performs some abstract task. You want to develop a user interface that will easily migrate. You also would like to deliver an application that will work on a PPC Macintosh. This article shows how this can be done.

TicTacPPC is an application that runs a particular Java program, *TicTacApp*. TicTacApp contains a call to a native function which is defined in C++ in TicTacPPC. From the perspective of the Java program, the C++ application is virtually the virtual machine. The application fulfills this role with the help of JManager.

TicTacApp

The CodeWarrior project, TicTacApp.java.π, creates a Java bytecode file, TicTacApp.zip. This sets up a Tic Tac Toe game on the screen. The user plays X and the program responds O.

The project has three files:

- TicTacApp.java which contains main().
- TicTacCanvas.java which handles the user interface.
- classes.zip, the Java libraries.

Since it contains a main()function, TicTacApp is a Java application. Since it refers to a native function, it can run only when that native function is defined and available to the Java runner.

TicTacPPC

The CodeWarrior project, TicTacPPC.π creates a Macintosh application, TicTacPPC which will run only if MRJ has been installed in the system. The folder containing TicTacPPC should also contain TicTacApp.zip.

TicTacPPC.π contains all the usual PowerPlant stuff plus MRJ stuff:

- JMSessionStubs.PPC
- NativeLibSupport.PPC

And the application specific files:

- CTicTacApplication.cp — the application object calls CJManager.cp to respond to New command.
- CFrameWindow.cp — support for Java AWT frames.
- CJManager.cp — communicates with the virtual machine through a JManager session and implements the native function.
- CTicTacEngine.cp — the class so smart that it never loses at TicTacToe.

The focus of this article is the work done by CJManager.cp and CFrameWindow.cp. CJManager.cp opens the file TicTacApp.zip and supports the native function. CJManager.cp is specific to this application.

Danny Swarzman develops software for fun and games. Fun for the users that is — he does it to earn money as a consultant. He also works to improve his game-playing skills at the San Francisco Go Club. He has been sited at www.stowlake.com. Send comments, questions and job offers to dannys@stowlake.com.

CFrameWindow.cp is relay service passing events between PowerPlant and JManager without regard for their contents. CFrameWindow.cp is a rudimentary version of a general class to support Java frames.

Figure 1 shows how the various pieces of this hybrid application fit together.

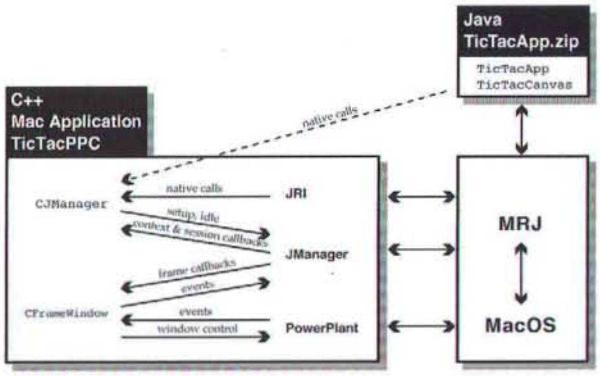


Figure 1. *How the pieces of this hybrid application fit together.*

Running Java Programs

Starting up the session

The application starts up the virtual machine by opening a *session* with JManager. The session is the structure through which JManager keeps track of the *Runtime Instance,* that particular virtual machine which will run our collection of threads of Java execution.

JManager uses a JMSession data structure to keep track of the session. The application has no access to the internals of the JMSession. It does provide JManager with a set of callback functions to handle standard files, stdin, stdout and stderr. The application also specifies security options, telling JManager how to limit what the Java program will be able access in the local system.

Since TicTacPPC will run only one Java program, TicTacApp, we don't worry about security and don't need to support standard files. All these are set to default values.

CJManager.cp

CJManager

```
The constructor sets up the JManager session.

CJManager :: CJManager ( LCommander *inSuperCommander )

// Set up a session with JManager. Setup a context for the frames.
{
   //This is an app that will run locally so security is not used.
   //To run apps, you might want to put sensible values here.
   static JMSecurityOptions securityOptions = {
        kJMVersion, eCheckRemoteCode,
        false, { 0 }, 0, false, { 0 }, 0,
        eUnrestrictedAccess, true };

   // If you want to implement standard files you must create functions
   // for stderr, stdout, stdin and put pointers to the functions into this
   // JMSessionCallbacks structure
```

Name Your Poison

Java OpenDoc (ODF) Tools Plus MPW PowerPlant Roaster CodeWarrior C++ Procedural C TCL SYMANTEC OOFILE neoAccess

AppMaker supports most popular languages and frameworks. Just point and click to design your user interface, then let AppMaker create resources and generate "human, professional quality code" to implement your design.

Use AppMaker as a prototyping and productivity tool or use it as a learning tool for Mac programming techniques or for new environments such as OpenDoc, Java, or PowerPlant.

AppMaker is just $299 and includes a one-year subscription on CD.

B•O•W•E•R•S Development

603-863-0945
bowersdev@aol.com
http://members.aol.com/bowersdev

```
static JMSessionCallbacks sessionCallbacks =
  { kJMVersion, nil, nil, nil };

// Create the session
ThrowIfOSErr_ ( JMOpenSession ( &sSession,
&securityOptions, &sessionCallbacks, 0 ) );

// Create the context for frames to support the AWT. These will be discussed later.
sContext = CFrameWindow :: CreateContext ( sSession,
  inSuperCommander );
}
```

Idling to give Java some time

The application gives the virtual machine time to service its threads by calling **JMIdle**. It is recommended that **JMIdle** be called at each cycle of the event loop. PowerPlant provides a convenient way to do that by subclassing from LPeriodical and overriding its SpendTime method.

CJManager.cp

SpendTime

The application calls this at idle time. It gives MRJ a chance to attend to its threads.

```
void CJManager :: SpendTime ()
{
  JMIdle ( sSession, kDefaultJMTime);
}
```

Finding Java Entities with JRI

The Java Runtime Interface is the standard for a C++ program to access Java entities used with MRJ 1.x. It was developed by Netscape to support code that works with their Navigator™ product. JRI allows the C++ program to find Java objects and contains specifications for conversion from Java types to C++ types.

The runtime stack and other data used by the virtual machine to keep track of the execution of a thread is the thread's *environment*. Calls to JRI pass an opaque structure representing the current environment. Through it, JRI locates objects, classes and methods.

Calling Java functions from C++

Through JManager calls, the application can virtually call Java functions. First the application uses JRI to locate the function and then uses JManager to invoke the function.

In RunApp() JManager is asked to execute the main() function of class TicTacApp in file TicTacApp.zip. First JManager calls are used to make the file available to the virtural machine. JRI calls locate the class. Finally the JManager call JMExecStaticMethodInContext() starts the process.

JRI specifies an encoding scheme to represent Java function signatures as strings. There are macros in JRI.h to construct them. Search for JRISig. Look at the macro definitions and the accompanying comments. You can infer the coding scheme, as is done here, or use the macros.

Actually JMExecStaticMethodInContext() tells the virtual machine to queue a request. TicTacApp's main() is not interpreted until the virtual machine gets around to it. The virtual machine runs when it is given time, that is when the application calls JMIdle().

Don't let your threads get tangled

Because the execution of the Java function is not immediate, the C++ program should not depend on the results being valid at a particular time. Deadlock will occur if the C++ program waits for a variable that is changed by the called Java function.

Multi-threaded or concurrent programming presents its own challanges. In this kind of application, there are extra opportunities for chaos. A good strategy would be to keep only one thread of C++ execution. Let the virtual machine manage multiple threads of Java. Keep the native functions short and fast.

CJManager.cp

CJManager

```
void CJManager :: RunApp ()
// Open file and call main in class appName.
{
  // Find the file.
  FSSpec fileSpec;
  JRIMethodID method;
  char *fileURL = "file:///$APPLICATION/TicTacApp.zip";
  ThrowIfOSErr_ ( JMURLToFSS ( sSession,
      fileURL, &fileSpec ) );
  ThrowIfOSErr_ ( JMAddToClassPath ( sSession, &fileSpec ) );

  // Find the class.
  JRIEnv* environment = nil;
  Assert_ ( environment = JMGetCurrentEnv ( sSession ) );
  JRIClassID appClass;
  char *appName = "TicTacApp";
  Assert_ ( appClass =
      JRI_FindClass ( environment, appName ) );
  // Run main. The third argument of JRI_GetStaticMethodID
  // specifies a signature of a Java function.

  // "([LJava/lang/String;)V" Specifies a function with
  // a single argument which is an array of references to objects
  // of class Java/lang/String. It returns type void.

  Assert_ (  method = JRI_GetStaticMethodID(environment,
      appClass, "main", "([LJava/lang/String;)V" ) );
  ThrowIfOSErr_ ( JMExecStaticMethodInContext( sContext,
  appClass, method, 0, nil) );
}
```

Providing Support for AWT

When the user does something, such as pressing a the mouse button, a chain of program activity starts. Here's what happens:

1. The user does something. The operating system reads the hardware and makes the information available for the next call to WaitNextEvent().
2. PowerPlant passes the event to the appropriate method in a class descended from a PowerPlant class. In our case it will be an event handler in CFrameWindow.
3. CFrameWindow passes the event to JManager.
4. JManager passes the event to the virtual machine which interprets the appropriate Java function.
5. The Java program responds to the event and creates visual feedback in a frame.
6. To provide the drawing environment for the Java frame, JManager calls callback functions in CFrameWindow.
7. The CFrameWindow callback manipulates the real windows with the help of PowerPlant.

The job of the application, handled by CFrameWindow, is to provide the event handler for step 3 and the callback for step 7.

Frames and windows

An object of the Java Class Frame is implemented in this application as a CFrameWindow object. CFrameWindow descends from the PowerPlant class, LWindow.

JManager passes a reference to a structure, JMFrameRef, to identify a frame. Through JManager calls, the application stashes a reference to its CFrameWindow inside the JMFrameRef structure. CFrameWindow maintains a pointer to finds its JMFrameRef.

Event handlers

Most events are passed on to JManager for the Java program to handle and respond as described above. For the activate and deactivate events, the event handler changes the appearance of the window itself because there is no provision for a callback to do it.

CFrameWindow.cp

DoSetBounds

This is called when the user resizes the window. It changes the bounds of the window and of the Java frame.

```
void CFrameWindow :: DoSetBounds ( const Rect &inBounds )
{
  JMSetFrameSize ( mFrame, &inBounds );
}
```

DrawSelf

When this is called, the window is being updated and the port is set up. It calls JManager to set up the process of drawing by the Java code.

```
void CFrameWindow :: DrawSelf ()
{
  JMFrameUpdate ( mFrame, GetMacPort()->visRgn );
}
```

HandleKeyPress

A key has been pressed when the window is in command. Forward the event to Java.

```
Boolean CFrameWindow :: HandleKeyPress( const EventRecord
&inKeyEvent)
{
  if ( inKeyEvent.modifiers & cmdKey ){
    JMFrameKey ( mFrame, inKeyEvent.message
    &charCodeMask, inKeyEvent.message >> 8,
  inKeyEvent.modifiers );
    return true;
  }
  else
    return false;
}
```

ObeyCommand

PowerPlant has detected a menu or key equivalent command when the window is in command. Forward the event to the Java program.

```
Boolean CFrameWindow :: ObeyCommand ( CommandT inCommand,
void *ioParam )
{
  switch ( inCommand )
  {
    case cmd_Close :
      JMFrameGoAway ( mFrame );
      return true;
  }
  return mSuperCommander->ObeyCommand ( inCommand, ioParam );
}
```

Bar Code Fonts & Readers

Create bar codes in any Windows or Macintosh program quickly & easily with Rivers Edge bar code fonts.

Bar Code Fonts

TrueType & Postscript Fonts
Easy to Use
30 Day, Money-Back Guarantee
$149 ea
Choose from:
Code 39, Code 93, Code 128, Int. 2 of 5, Codabar, MSI-Plessey, PostNet, UPC/EAN/ISBN

Bar Code Readers

Attaches to Keyboard
Plug & Play for Windows/Mac
Easy to Use
$229 - Wand Reader-Basic
$295 - CCD Reader
$495 - Laser Reader

0 12345 67890 5

riversedge
bar coding made simple™

Call for FREE Information

512.219.7768

fax: 512.219.7769 **email:** sales@riversedge.com

www.riversedge.com

ClickSelf

PowerPlant has detected a click in the active window. Forward the event to the Java program.

```
void CFrameWindow :: ClickSelf ( const SMouseDownEvent
&inMouseDown )
{
  JMFrameClick ( mFrame,
    inMouseDown.whereLocal,
    inMouseDown.macEvent.modifiers );
}
```

ActivateSelf

This is called when an activate event is received by the window. The Java frame is activated and the window is activated.

```
void CFrameWindow :: ActivateSelf ()
{
  JMFrameActivate ( mFrame, true );
  LWindow :: ActivateSelf ();
}
```

DeactivateSelf

This is called when an deactivate event is received by the window. The Java frame is deactivated and the window is deactivated.

```
void CFrameWindow :: DeactivateSelf ()
{
  JMFrameActivate ( mFrame, false );
  LWindow :: DeactivateSelf ();
}
```

Frame callbacks

JManager calls these to do the actual work for the Java Frame object. They are declared static.

FindFrameWindow

Retrieve the reference to the CFrameWindow object from the client data field of the frame structure.

```
CFrameWindow *CFrameWindow ::
    FindFrameWindow ( JMFrameRef frame )
{
  CFrameWindow *result = nil;
  if ( frame )
    if ( JMGetFrameData (
        frame, (JMClientData*) &result ) == noErr )
      return result;
  return nil;
}
```

SetupPortCallback

This frame callback sets the port for drawing. The application can use the return value of this function to pass an old port reference that can be later retrieved by RestorePortCallback() as a form of client data. The value is given back to the application in the callback to restore the port. This application doesn't need to do this.

```
void *CFrameWindow :: SetupPortCallback ( JMFrameRef frame )
{
  OutOfFocus ( nil );
  CFrameWindow *window = FindFrameWindow ( frame );
  if ( window )
    window->FocusDraw();
  return nil;
}
```

RestorePortCallback

This callback is provided so that the application can save data, like a port, with the setup callback and restore it here. This application doesn't do that.

```
void CFrameWindow :: RestorePortCallback (
    JMFrameRef /*frame*/, void */*param*/ )
{
}
```

ResizeRequestCallback

This frame callback resizes the window.

```
Boolean CFrameWindow :: ResizeRequestCallback
    ( JMFrameRef frame, Rect *desired )
{
  CFrameWindow *pane = FindFrameWindow ( frame );
  if ( pane && desired )
  {
      Rect r = pane->mUserBounds;
      r.bottom = r.top + desired->bottom - desired->top;
      r.right = r.left + desired->right - desired->left;
      pane->LWindow :: DoSetBounds ( r );
    return true;
  }
  return false;
}
```

InvalRectCallback

This frame callback marks a rectangle as needing to be updated.

```
void CFrameWindow :: InvalRectCallback ( JMFrameRef frame,
const Rect *r )
{
  CFrameWindow *pane = FindFrameWindow ( frame );
  if ( pane )
    pane->InvalPortRect ( r );
}
```

ShowHideCallback

This frame callback shows or hides the window.

```
void CFrameWindow :: ShowHideCallback ( JMFrameRef frame,
  Boolean showFrameRequested )

{
  CFrameWindow *pane = FindFrameWindow ( frame );
  WindowPtr window = pane->GetMacPort();
  if ( pane )
    if ( showFrameRequested )
      ShowWindow ( window );
    else
      HideWindow ( window );
}
```

SetTitleCallback

This frame callback changes the window title.

```
void CFrameWindow :: SetTitleCallback ( JMFrameRef frame,
Str255 title )
{
  CFrameWindow *pane = FindFrameWindow ( frame );
  if ( pane )
    pane->SetDescriptor ( title );
}
```

CheckUpdateCallback

If the update region isn't empty start the update process.

```
void CFrameWindow :: CheckUpdateCallback ( JMFrameRef frame )
{
  CFrameWindow *pane = FindFrameWindow ( frame );
  if ( window && !EmptyRgn(
      ((WindowPeek)(window>GetMacPort()))->updateRgn))
    window->UpdatePort();
}
```

Creating and destroying frames

When the virtual machine needs to create a new frame, JManager calls an application callback function to create the Macintosh structures needed for the frame. A group of functions is identified to JManager as a *context*. These functions create and destroy frames and provide for exception notification.

TicTacPPC maintains only one context. The callback functions are declared as static in CFrameWindow. In addition to identifying the context callbacks, the application can store data in a field of JManager's context structure, referenced by a JMAWTContextRef.

In this application, the client data of the JMAWTContextRef structure is used to store a reference to the commander object which will eventually be the super commander of the CFrameWindow objects used for frames. Later, RequestFrameCallback() will use the commander object to create a new CFrameWindow.

CFrameWindow.cp

CreateContext

Create a context using our context callbacks. Put the reference to the super commander into context structure as client data.

```
JMAWTContextRef CFrameWindow :: CreateContext ( JMSessionRef
inSession,
  LCommander *inSuperCommander )
{
  static JMAWTContextCallbacks contextCallbacks =
  {
    kJMVersion, // always this constant.
    RequestFrameCallback,
    ReleaseFrameCallback,
    UniqueMenuIDCallback,
    nil // No exception handling - you may want to add it.
  };
  JMAWTContextRef context;
  ThrowIfOSErr_ ( JMNewAWTContext ( &context, inSession,
      &contextCallbacks, 0 ) );
  ThrowIfOSErr_ ( JMSetAWTContextData ( context,
  (JMClientData)inSuperCommander ) );
  ThrowIfOSErr_ ( JMResumeAWTContext ( context ) );
  return context;
}
```

RequestFrameCallback

This context callback creates a new CFrameWindow for the new frame. This implementation ignores all the characteristics requested in the call because it will be used only with one particular Java program. To make this function more general, use these to set the window parameters.

```
OSStatus CFrameWindow :: RequestFrameCallback (
  JMAWTContextRef context, JMFrameRef newFrame,
  JMFrameKind /* kind */, UInt32 /*width*/,
  UInt32 /*height*/, Boolean /* resizable */,
JMFrameCallbacks *callbacks )
{
  callbacks->fVersion = kJMVersion;
  callbacks->fSetupPort = SetupPortCallback;
  callbacks->fRestorePort = RestorePortCallback;
  callbacks->fResizeRequest = ResizeRequestCallback;
  callbacks->fInvalRect = InvalRectCallback;
  callbacks->fShowHide = ShowHideCallback;
  callbacks->fSetTitle = SetTitleCallback;
  callbacks->fCheckUpdate = CheckUpdateCallback;

  //The context client data contains a reference to a LCommander object.
  JMClientData data;
  JMGetAWTContextData ( context, &data );
  CFrameWindow *window = (CFrameWindow*)CreateWindow (
    kFrameWindowResID, (LCommander*)data );

  // Identify the frame structure with the window.
  window->mFrame = newFrame;
  //The frame's client data points to the window.
  JMSetFrameData ( newFrame, (JMClientData*)window );
  window->Show();
  return noErr;
}
```

ReleaseFrameCallback

JManager is done with the frame. Destroy its CFrameWindow object

```
OSStatus CFrameWindow :: ReleaseFrameCallback (
JMAWTContextRef /* context */, JMFrameRef oldFrame )
{
  CFrameWindow *pane = FindFrameWindow ( oldFrame );
  delete pane;
  return noErr;
}
```

UniqueMenuIDCallback

This context callback isn't used because the Java app that we're running doesn't create any menus. This code was copied from Apple sample code.

```
SInt16 CFrameWindow :: UniqueMenuIDCallback ( JMAWTContextRef
    /*context*/, Boolean isSubmenu )
{
  static SInt16 theFirstHierMenu = 1;
  static SInt16 theFirstNormalMenu = 500;
  if (isSubmenu )
    return theFirstHierMenu++;
  return theFirstNormalMenu++;
}
```

Implementing a Native Function

The Java class TicTacCanvas contains an interface for a function:

static native void DoOMove(char[]board);

The keyword native tells the compiler that the function is defined by the local system. In this case, it is defined in the application.

A C++ function will take an Java array representing the position on the board when it is O's turn to play. After the native function executes, the array will contain the new O move.

The C++ program must tell JManager which function will implement the native. To identify the Java funtion, the program uses signatures as discussed in the JRI documentation. The signature for this function is "DoOMove([C)V".

TestTrack

Your Total Bug Tracking Solution

Powerful enough to handle all your bug tracking needs. **Affordable** enough to put on everyone's desktop.

Introducing TestTrack 1.5 and Solo Bug!

Discover the tool more and more of today's top software developers are using to improve the quality of their Mac and Windows applications—**TestTrack**.

- Track bugs, feature requests, customer information, and more.
- Use on Mac or Windows 95/NT platforms.
- Produce concise reports.
- Link engineers, testers, managers, even tech writers.
- Automatically route bugs to team members.
- Save time and improve tech support by giving **Solo Bug**, our stand-alone bug reporter, to your customers (comes free with TestTrack 1.5).
- Track the history of each bug.
- Multiple users, full security!
- Ideal for consultants too!

To order call 513-683-6456

Seapine Software, Inc. 1066 Seapine Ct. Maineville, OH 45039

sales@seapine.com
http://www.seapine.com

Seapine Software

CJManager.cp

RegisterNative

Identify CJManager::DoOMove() as the C++ function that handles the Java native call to TicTacCanvas.DoOMove().

```
void CJManager :: RegisterNative ()
{

  // Find the class.
  JRIEnv* environment = nil;
  Assert_ ( environment = JMGetCurrentEnv ( sSession ) );
  JRIClassID canvasClass;
  static char *canvasClassName = "TicTacCanvas";
  Assert_ ( canvasClass = JRI_FindClass ( environment,
      canvasClassName ) );

  // Create signatures for all the native functions. We have only one function.
  //The signatures include the function name. This function
  // passes one argument which is an array of the Java type char. It returns void.

  static char *signatures = "DoOMove([C)V";

  //To support the one native function, there is one C++ function.
  // Pass a pointer to an array with one element.

  static void *procArray[] = { DoOMove };
  JRI_RegisterNatives ( environment, canvasClass,
      &signatures, procArray );

}
```

The implementation of the native function

The array passed to JRI_RegisterNatives contains pointers to functions defined as:

```
typedef void (*JRI_NativeMethodProc)(JRIEnv* env,
    jref classOrObject, ...);
```

The first parameter identifies the thread of Java execution invoking the function. The second is the Java object for which the function is called. If the function is a class function, the second parameter is the class for which the function is defined.

Succeeding parameters are the parameters in the original Java call. Each of these is of type jref, the general-purpose JRI type.

In the case of Java function, DoOMove(), there is one parameter which is a reference to a Java array of Java type char. The type jchar is defined in JRI to represent Java type char. The reference to a Java array is not the same as a pointer. The Java specifications say that an array of a primitive type is represented as a series of contiguous storage locations. The actual data is located somewhere in the stack of the Java thread. For this purpose, DoOMove must call GetScalarArrayElement().

Included in Sun's JDK there is a utility, javah, to help set up prototypes for native functions. Using it is more work than writing your own prototype for one function. It will be obsolete with the next version of Java.

DoOMove

This function supports the Java native function void DoOMove(char[]ioBoard);

Find the pointer to the data in an array of java char. Call the engine to make the move in the C++ array whose elements are jchar.

```
void CJManager :: DoOMove ( JRIEnv *env,
    jref /*JavaObject*/, jref ioBoard )
{
  jchar *board = (jchar*) env->GetScalarArrayElements (
ioBoard );
  CTicTacEngine :: BestOPossible ( board );
}
```

The Application Class

CTicTacApplication is an PowerPlant LApplication subclass. It calls CJManager to start up, to spend idle time and to respond to New menu commands.

CTicTacApplication.cp

CTicTacApplication

Register the class PowerPlant class which handles Java frames. Start up JManager. Start recieving idle events.

```
CTicTacApplication :: CTicTacApplication()
{
  RegisterClass_(CFrameWindow);
  CJManager startupASession ( this );
  StartIdling();
}
```

SpendTime

Forward idle event to JManager. Overrides LPeriodical function.

```
void CTicTacApplication :: SpendTime()
{
  CJManager :: SpendTime();
}
```

MakeNewDocument

Respond to New.

```
LModelObject *CTicTacApplication :: MakeNewDocument()
{
  CJManager :: OpenApp ();
  CJManager :: RegisterNative();
  return nil;
}
```

Building and Debugging The Application

First install all the MRJ stuff and read the MRJ docs and the JRI docs.

To build TicTacPPC, start with the usual PowerPlant stuff.

Add MRJ libraries to the project. For PPC they're JMSessionStubs.PPC and NativeLibSupport.

Add access paths for the includes in the MRJ SDK.

Add an access path for the Metrowerks Standard Library C includes.

Set in the C/C++ Language settings "Enums Always Int".

This line in JRI.h causes a problem:

```
void Throw(JRIThrowableID throwableID)
  { interface->ThrowProcPtr(this, throwableID); }
```

I commented it out. You may choose a more elegant solution, especially if you want your callback to be able to throw a Java exception.

Build the project and start debugging. Debug the application in the usual way. If you need to step through the Java app at the same time, there is an extra step. Open the .zip file in the debugger and set a breakpoint where you want it. Now you can debug the application in the usual way. It will stop at your Java breakpoint as well as those in the application.

Conclusions and Future Directions

MRJ, in conjunction with CodeWarrior and PowerPlant provides an excellent environment for developing an application with parts in Java and parts in C++.

There are some pitfalls for developing larger projects. The most apparent problem is the delay in the sequence of upgrades in the long trek from Mountain View to Cupertino. It gets to Washington several months earlier. MRJ 2.0 corresponding to Sun's SDK 1.1 lags behind other platforms by many months. There's the additional lag for a version for 68k Macs. This is pioneering stuff and it is reasonable to expect that there be some retrofitting between the beginning of development and release time for a product using MRJ.

If you application permits, it would be best to confine the interface between the C++ application and the Java code to something very simple. Here we just invoke the main() and let the Java program take it from there. Instead of making many prototypes and signatures, implement one native and avoid the mess. The bulk of the work that you put into a project of this nature will endure.

Bibliography and References

The MRJ package is available from Apple's web site at <http://appleJava.apple.com/>.

Documentation on the Java Runtime Interface (JRI) is available at <http://home.netscape.com/eng/jri/>.

Another example of using PowerPlant with MRJ: <http://www.fullfeed.com/~lorax/powerplant.html>.

Credits

Thanks to Victoria Leonard for the artwork. Thanks to Mark Terry and Bob Ackerman for reviewing work in progress.

by Bob Boonstra, Westford, MA

IMAGE LOCATOR

Imagine yourself with a collection satellite images and the task of finding a particular item in those images. Rather than look for a needle in this image haystack manually, you might call on your PowerMac to narrow down the choices for you. We'll enlist the aid of our Programmer's Challenge participants to help you do that job quickly. The Challenge this month is to detect the presence of a target pattern inside a larger background image. Because the background has been detected by an imperfect sensor, there is noise present in the background image. Your code will need to detect the target in the background despite this noise.

The prototype for the code you should write is:

```
#define topLeft(r)      (((Point *) &(r))[0])

void InitTarget(
  BitMap pattern,     /* image to be detected */
  BitMap mask         /* bits in image that we care about */
);

long /* numFound */ ImageDetect(
  BitMap backgroundImage,  /* find the target image in backgroundImage */
  Point locations[],       /* return topLeft of matching locations here */
  long maxLocations,       /* max number of locations to return */
  float noise              /* allow this fraction of mismatched bits */
);

void CleanUp(void); /* deallocate any memory allocated by InitTarget */
```

Image location will take place in two steps. First, your InitTarget routine will be called with the target pattern that you will be looking for. Next, the ImageDetect routine will be called multiple times with a different background image and an associated noise threshold. ImageDetect should locate all occurrences of the target in the background, allowing for mismatched bits up to the noise threshold, and return the location of the pattern matches. Finally, the CleanUp routine will be called to allow you to deallocate any memory allocated by InitTarget.

InitTarget will be called with two BitMaps that describe the target pattern to be detected. The pattern BitMap identifies bits that should be set to 1, and the mask BitMap describes the bits that you care about (1s and 0s). Any bits not in the mask are not part of the target image, and the corresponding values in the background image are to be ignored. InitTarget should process the target image as desired and allocate memory to remember it.

ImageDetect will then be called multiple times (5-10 on average for each call to InitTarget). You should locate each occurrence of the target image in backgroundImage and return the coordinate in the background of topLeft(pattern.bounds) in the locations array. The noise parameter describes the fraction of target bits where the backgroundImage is allowed to differ from the target and still be considered a match. Up to noise times the number of 1s in the mask, rounded down, bits may be mismatched. Normally, locations will be large enough to hold all of the matches found, but you should not return more than the maxLocations matches for which storage has been allocated. The pattern matches may be returned in any order. If the maxLocations limit is exceeded, the choice of which matches to report is yours. ImageDetect should return the number of matches found.

Other information: The bounds rectangle for the pattern and the mask will be identical. All bits set in the pattern will also be set in the mask (but not the converse). The backgroundImage will typically be the size of a large monitor (e.g., 1024x768, or 1600x1200).

This will be a native PowerPC Challenge, using the latest CodeWarrior environment. In keeping with tradition, September is assembly language month here at the Programmer's Challenge. Solutions may be coded in PowerPC or 68K assembly language, C, C++, or Pascal.

Finally, we should note that the Programmer's Challenge began its sixth year last month. During that time, the Challenge has changed development environments, moved from 68K to PowerPC, and expanded its selection of languages. We appreciate the participation of our readers, without which the Challenge would not be possible. Happy belated birthday, Programmer's Challenge.

THREE MONTHS AGO WINNER

The June Challenge was to implement a Turing Machine, a finite state machine augmented with an infinite amount of external storage. Twenty people submitted entries, and 17 of those worked correctly. Congratulations to **Ernst Munter** (Kanata, Ontario) for submitting the fastest solution and returning to the Challenge winner's circle.

The key to success in this Challenge was being able to quickly find the rule that applied to the current machine state and the current input symbol. A variety of techniques were used to find the applicable rule. Hashing was used by many of the faster entries. Ernst uses either hashing or a simple lookup table, depending on memory availability. Others sorted the rules and used a binary search. The slower solutions typically used a brute force approach of simply searching linearly through the rule set.

I used two types of test cases to stress the solutions. The first case involved a Turing Machine of approximately 2300 rules that sorted an input tape with an alphabet of 30 symbols and tape lengths of about 100 symbols. This case required over 113,000 Turing Machine state changes. The second test case was a Universal Turing Machine. A UTM is an interesting creature. Its input tape has two parts, an encoded version of the rules (program) for another Turing Machine, which it is to execute, and the input tape for that emulated program. The tape also contains an area where the Universal TM maintains the state for the program being emulated. The UTM operates by looking up the rule (or program instruction) that applies given the current state of the machine being emulated, remembering that instruction while it moves to the current input for the emulated machine, and then executing that instruction. The Universal Turing Machine I used operated on a binary alphabet and consisted of 184 rules, operating on an input tape that described a simple unary addition machine. This test case required just under 240,000 state changes to execute.

The table below lists for each entry the execution times in milliseconds for the sort test case and the Universal Turing Machine case, total execution time, code and data sizes, and the programming language used. The number in parentheses after the entrant's name is the total number of Challenge points earned in all Challenges to date prior to this one.

Name	Time1	Time2	Total Time	Code	Data	Language
Ernst Munter (246)	28.1	29.3	57.6	920	8	C++
Russ Webb	30.9	33.0	64.2	1696	140	C
Devon Carew	33.4	35.6	79.5	976	28	C
Gary Beith (24)	39.6	39.9	86.9	592	32	C
Mason Thomas (4)	47.9	40.5	89.1	740	8	C
Kevin Cutts (57)	42.4	43.9	89.7	620	32	C++
Simon Holmes à Court	44.1	42.3	90.0	744	32	C++
Juerg Wullschleger	48.5	50.4	99.7	476	8	C
Daniel Harding	96.0	63.0	159.7	2612	406	C++
Zach Thompson	93.2	64.8	161.3	1076	48	C++
Gregory Cooper (54)	107.8	84.6	192.7	668	40	C
Graham Herrick	137.0	107.2	244.8	820	16	C
Andy Scheck (17)	3239.0	158.3	3397.0	212	8	C++
Charles Higgins (20)	3238.0	167.8	3406.0	276	8	C
David Whitney	4174.0	272.3	4449.0	19800	2745	C++
Bjorn Davidsson (6)	6565.0	165.6	6731.0	224	8	C++
Terry Noyes	6737.0	198.3	6936.0	200	8	C
R.B.				2736	99	C
S.A.				840	448	C
W.R.				1148	8	C++

TOP 20 CONTESTANTS

Here are the Top Contestants for the Programmer's Challenge. The numbers below include points awarded over the 24 most recent contests, including points earned by this month's entrants.

Rank	Name	Points	Rank	Name	Points
1.	Munter, Ernst	196	11.	Nicolle, Ludovic	21
2.	Gregg, Xan	83	12.	Picao, Miguel Cruz	21
3.	Cooper, Greg	54	13.	Brown, Jorg	20
4.	Lengyel, Eric	40	14.	Day, Mark	20
5.	Boring, Randy	37	15.	Gundrum, Eric	20
6.	Lewis, Peter	32	16.	Higgins, Charles	20
7.	Mallett, Jeff	30	17.	Slezak, Ken	20
8.	Murphy, ACC	30	18.	Studer, Thomas	20
9.	Larsson, Gustav	27	19.	Karsh, Bill	19
10.	Antoniewicz, Andy	24	20.	Nevard, John	19

There are three ways to earn points: (1) scoring in the top 5 of any Challenge, (2) being the first person to find a bug in a published winning solution or, (3) being the first person to suggest a Challenge that I use. The points you can win are:

1st place ....20 points	5th place ....2 points
2nd place ......10 points	finding bug ....2 points
3rd place ....7 points	suggesting Challenge...2 points
4th place........4 points	

Here is Ernst's winning solution:

Turing.cp ® 1997 Ernst Munter

```
                                                   Problem Statement
Implement the engine for a Turing Machine, a state machine
  which, at each step, reads a symbol from a tape, consults
  a rule which is a function of the current state and the
  symbol. The rule specifies a new state, a symbol to output,
  and the direction in which to move the tape, or to halt.

                                                            Solution
I first try to build a lookup table as an index into the
  rules array. But this may require an "unreasonable"
  amount of memory.

  First, I scan the rules to determine the amount of table
  memory required for a simple lookup index. If this
  appears to be too much, I go to plan B: a hashed index.

  The hash table uses linear open addressing: when the table
  is built and an index location is needed which is already
  in use we have a collision. To resolve it, I scan linearly
  through the index array until a free location is found.
  The size of the index array is larger than the number of
  rules, so a free location will always be found.

                                   Optimization of hash table lookup
  The majority of rules will hash to unique index addresses.

  Rules which hash to the same value can be seen as a
  sequence of index table entries, with the primary location
  containing the first rule address to be found.

  When the Turing Machine is executing, any colliding rule
  that is encountered will have its index moved to the
  primary index location, on the assumption that it will be
  used again, and will then be found more quickly.

                                                         Assumptions

  A minimum amount of table memory of 8K entries is always
  provided. But for larger rule sets, an index table that
  will occupy about 1/2 to 3/4 the amount of memory taken
  by the rules array itself may be allocated.

  The memory allocated for the simple lookup table will be
  the number_of_states * number_of_symbols, rounded up to
  a power of 2, but not more than 8K entries,
  or 2 * number_of_rules, whichever is larger.

  If the memory required for the simple index would exceed
  those rules, for example if there are a lot of holes
  in the symbol/state space, the hashed index is used.

  The memory allocated for the hash index array will
  be the larger of 8K entries or 2 * number_of_rules, rounded
  up to the nearest power of 2.

  For example, a rule set of up to 2K rules may result in
  a 32K byte index; a rule set of 50,000 rules which occupy
  1M of rules memory may get an index table of 512K bytes.

#include <stdio.h>
```

TCP/IP Scripting Addition — *The Internet Scripting Solution*

The **TCP/IP Scripting Addition** allows you to quickly develop Internet client/server applications using AppleScript®. If you want to script with MacTCP™ and Open Transport™, here's your solution!

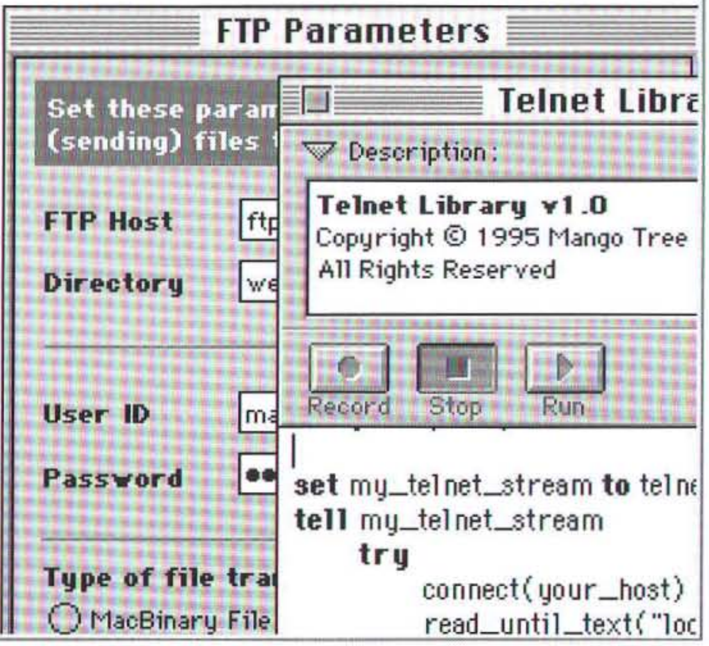
- Supports ScriptEditor, FaceSpan™, and HyperCard™
- Build net–wise WebSTAR ™CGI scripts and custom web clients
- Sample scripts include FTP, HTTP, Telnet, Post Office, E–Mail and more
- Featured on the Apple® Internet Server

ONLY $49 SRP

Order now through Developer Depot at 805–494–9797 or other mail order stores

Mango Tree Software, Inc.
Box 1057 • Brookline, Massachusetts USA 02146
Fax / Voice 617–327–8663
sales@mangotree.com • www.mangotree.com

All trademarks are properties of their respective holders.
Contact Mango Tree Software for site licensing and redistribution information.
Copyright © 1993–7 Mango Tree Software, Inc.



Telnet Library v1.0
Copyright © 1995 Mango Tree
All Rights Reserved

```
set my_telnet_stream to telne
tell my_telnet_stream
    try
        connect( your_host)
        read_until_text("loc
```

```
#include <stdlib.h>
#include <string.h>
#include "turing.h"

const enum {    // constants controlling min size of index
  EXPANSION  = 2,
  MIN_BITS   = 13,
  MIN_SIZE   = 1L<<MIN_BITS };

typedef const TMRule* TMRulePtr;

Boolean TuringMachine(
  const TMRule theRules[],
  ulong numRules,
  ulong *theTape,
  ulong tapeLen,
  long  rwHeadPos,
  TMMoveProc ReportMove
) {
  TMRulePtr* index;
  ulong    state=0;
  ulong    symbol;
  int    direction;
  ulong* tape=theTape+rwHeadPos;
  ulong* tapeEnd=theTape+tapeLen;

  ulong    mask;
  TMRulePtr rule=theRules;

//The function contains 2 very similar sections,
// one section uses a plain lookup table for an index,
// the other uses a hash table.

//Try to construct a collision-free index of rule addresses

// compute table size
  ulong  maxState=rule->oldState;
  ulong  maxSym=rule->inputSymbol;
  ulong  minSym=rule->inputSymbol;
  rule++;
  for (int i=1;i<numRules;i++) {
    if (maxState<rule->oldState) maxState=rule->oldState;
    if (maxSym<rule->inputSymbol) maxSym=rule->inputSymbol;
    else
    if (minSym>rule->inputSymbol) minSym=rule->inputSymbol;
    rule++;
  }
  ulong numSyms=maxSym-minSym+1;
  ulong numStates=maxState+1;
  ulong numIndex=(numStates)*(numSyms);

  if ((numIndex<numStates)                     // overflow
    || ((numIndex > MIN_SIZE)
    && (numIndex > EXPANSION*numRules)))  // too large
      goto try_hash;

// increase size to the next power of 2
  ulong dummy=1;
  while (numIndex) {
    numIndex>>=1;
    dummy<<=1;
  }
  numIndex=dummy;

// Allocate the table memory
  index=(TMRulePtr*)malloc(numIndex*sizeof(TMRulePtr));

// Always expect to get the memory, but just in case ...
  if (index==0)
    return FALSE;

// All unused index locations will remain 0
```

```
    memset(index,0,numIndex*sizeof(TMRulePtr));
    mask=numIndex-1;

// Scan the rules and populate the index array
    rule=theRules;
    for (int i=0;i<numRules;i++) {
      ulong addr=
        mask & (rule->oldState*numSyms+rule->inputSymbol);
      index[addr]=rule++;
    }
// Using the collision-free index table:
// Loop until the tape halts or we fail on error
    do {
        symbol=*tape;
        ulong addr=mask & (state*numSyms+symbol);
        rule=index[addr];
        if (rule == 0)      // illegal symbol, no rule
          break;
        symbol=rule->outputSymbol;
        state=rule->newState;
        direction=rule->moveDirection;

        ReportMove(symbol,state,MoveDir(direction));
        *tape=symbol;

        if (direction==kHalt)    {
          free(index);
          return TRUE;
      }

        tape+=direction;
    } while ((tape>=theTape) && (tape<tapeEnd));
    free(index);
    return FALSE;

try_hash:
// Section 2
// If we get here, we could not make a simple table
// and have to go with a hash table, collisions are possible

// Find table size >= minimum size
    numIndex=MIN_SIZE;
    while (numIndex < EXPANSION*numRules) {
      numIndex*=2;
    }

// Allocate the table memory
    index=(TMRulePtr*)malloc(numIndex*sizeof(TMRulePtr));

// Always expect to get the memory, but just in case ...
    if (index==0)
      return FALSE;

// All unused index locations will remain 0
    memset(index,0,numIndex*sizeof(TMRulePtr));

    mask=numIndex-1;
    ulong  hFactor=1 | (numIndex/numStates);
    long      hDelta=1 | (hFactor>>1);

// Scan the rules and populate the index array
    rule=theRules;
    for (int i=0;i<numRules;i++) {
      ulong addr=
        mask & (rule->oldState*hFactor+rule->inputSymbol);

// if primary location is not empty: find next free location
      while (index[addr])
        addr=mask & (addr+hDelta);
      index[addr]=rule++;
    }

// Using the hash index table:
// Loop until the tape halts or we fail on error.
// This loop is the same as the loop in the first section
// except we have to check for possible collisions with each rule..
    do {
        symbol=*tape;
        ulong addr=mask & (state*hFactor+symbol);
        rule=index[addr];
        if (rule == 0)            // illegal symbol, no rule
          break;

// check if we have the right rule, or a collision
        if ((symbol != rule->inputSymbol)
          || (state != rule->oldState)) {
          const TMRule* rule0=rule;
            ulong addr0=addr;

          do {            // resolve the collision
            addr=mask & (addr+hDelta);
            rule=index[addr];
              if (rule == 0) {        // could not find rule
            free(index);
          return FALSE;
          }

          } while ((symbol != rule->inputSymbol)
          || (state != rule->oldState));
          index[addr]=rule0;          // move last-used rule
          index[addr0]=rule;          // up in chain

        }

// now we have the correct rule
          symbol=rule->outputSymbol;
          state=rule->newState;
          direction=rule->moveDirection;

          ReportMove(symbol,state,MoveDir(direction));

          *tape=symbol;
          if (direction==kHalt) {     // normal stop
            free(index);
            return TRUE;
            }

            tape+=direction;
    } while ((tape>=theTape) && (tape<tapeEnd));

    free(index);
    return FALSE;
}
```

by Dave Mark, ©1997 by Metrowerks, Inc., all rights reserved.

A CodeWarrior Java Update

Tim Freebill *is the Engineering Manager for the Java Tools team, with technical responsibilities for the compiler, linker and preference panel plugins. He also concentrates on developing his pointy hair style.*

Greg Bolsinga *writes Class Wrangler, worked on porting the 1.1.1 javac compiler, and enjoys relieving the TX heat at the local swimming holes while not coding. He is a Scorpio, and his favorite color is purple.*

Clinton Popetz *just can't keep his hands out of the Java pot, and has the burns to prove it. When not engaged in mind-meld with his Mac, he is a typical slap-happy father of two, and a practioner of Kuk Sool Won, a martial art in which he serves as a chew toy for evil black belts.*

Scott Kovatch *is a software engineer who lives in Round Rock, Texas with his wife Sandy, and their two cats Star and Shadow. He has been working on Macintosh applications for five years, and is currently working on the native implementations for the Metrowerks 1.1 Java virtual machine and Metrowerks Java.*

Kevin Buettner *is the JIT compiler lead. He concentrates on the overall JIT architecture, performance enhancements, and the PowerPC code generator.*

Laurent Visconti *is a "virtual" Metrowerks engineer living in Boston. He spans the bridge between the Java and compiler groups at Metrowerks while working on the Java bytecode compiler and the Java to native compiler.*

It's hard to believe, but it was just last year that Java first made its appearance on the Mac. Since then, lots has changed. This month's interview is with the Metrowerks Java team. Read on to learn what's coming down the pike in CodeWarrior Java...

Dave: What Java related technologies is Metrowerks working on these days?

Tim: CodeWarrior Java has seen a lot of extra attention recently, and we've really put a lot of work into streamlining the tools and expanding the Java support within CodeWarrior. We are making CodeWarrior work better as a Java development environment, and we're adding new functionality in several areas.

For example, we are working on our ability to compile Java source directly into machine code, leveraging our existing compiler backend technologies to produce quality Java to native code for a variety of platforms. We'll also have a new backend which uses our optimization technology to produce highly optimized bytecodes. Our ClassWrangler tool is growing by leaps and bounds, giving developers a more sophisticated mechanism for exploring their class libraries. Constructor for Java will be growing into a full Rapid Application Development tool for Java and will be fully integrated into the IDE. Lastly, our VM has received perhaps the most attention, giving us current JDK compatibility and undergoing many, many optimizations for speed, both in the core VM and in the JIT.

Dave: What's the story with the CodeWarrior Java to Native compiler? Will it be compatible with the various CW Back-ends?

Laurent: The Java front-end will generate the standard CodeWarrior IR (Intermediate Representation) that is used by all the CW back-ends. Therefore the Java compiler will be able to invoke any existing or future CW back-end. Because of this architecture developers will be able to generate native binaries with no need for a JIT or an interpreter. This architecture also enables the various optimizations performed both at the IR level and in the back-ends. This will allow the generation of binary code from Java sources equivalent in quality to code generated from C/C++ sources.

Dave: And the new Java back-end?

Laurent: We will implement a new Java back-end that will translate IR to Javabytecode. Because of the optimizations performed on the IR, this new back-end will allow developers to generate better optimized bytecodes.

Tim: This is the same optimization technology CodeWarrior uses in some of its other compilers, so it's proven. It will make even the interpreted bytecodes run more quickly, even without a JIT; with a JIT in place, the code runs that much faster. Also, size optimization can make applet class files download more quickly.

Dave: What's new with Class Wrangler?

Greg: First, Class Wrangler is a program that will allow Mac users to create zipfiles and examine zip files, with an emphasis on zip archive files for Java. You can view all the classes in your archive, and add and remove classes from the archive. As Java programmers already know, it's really easy to get class files with file names longer than the Mac's limit of 31 characters.

When you drop a class file onto Class Wrangler, it will examine the actual class data to determine what the class really should be called in the zip archive. Class files keep lots of data about themselves in their object code. In the Information window, you can see all the methods, all the data members, the super class, all the interfaces, the source file, and the package of a class. You can copy the java code or the java signature to the clipboard, which really comes in handy for native method calls, both Java->native and native->Java.

Class Wrangler is also ready for JDK 1.1's JAR files. It will read and write compressed JAR files. We will be expanding ClassWrangler's JAR capability to handle java digital signatures and other security features. Soon we'll also add the ability to put any type of file within the zip file's hierarchy. This way users can create an images directory for their gifs for JDK 1.1, all within a single JAR file. Class Wrangler is a simple utility for Mac Java programmers, but it's handy for all Java programmers. It will be on Windows soon.

Dave: What class libraries will CodeWarrior support?

Tim: Java frameworks are moving towards the component model, which is great for RAD. By using a defined component API, such as Java Beans, we will be able to support a variety of frameworks. The tool need only know the component API. It can query the component, and the user can put them together in a meaningful fashion. This allows our users to choose what framework they want to use, and lets us remain agnostic about whose framework we support.

Dave: Constructor for Java has been final or some time now. Can you tell me about the next generation RAD tools as they relate to Java?

Clint: The preferred mechanism for RAD in Java is quickly becoming Java Beans. Beans provides a simple metadata infrastructure for component software, and this makes it very attractive to RAD tool designers. In the past, developers created their own proprietary scheme for managing processes such as reanimation, event hookup, component introspection, description, and packaging. Beans provides a common API for these pieces, and provides it in a framework independent fashion. Currently, the Java framework landscape is diverse and confusing. But by building our Java RAD plans on Beans, our work can go forward as the framework wars are waged, without fear of obsoletion.

Dave: What VMs will CodeWarrior support? Apple MRJ? Sun JDK VM for Mac? Microsoft VM?

Scott: Well, there are a couple of levels of 'support'. The first is bytecode support — our Java compiler generates bytecode that will work with any Java VM, whether it's on Windows, Mac, Unix, or any other platform. Then there is run support, which means "What can CodeWarrior tell to run your Java code?" This one's pretty broad as well — just about any Java-enabled browser, Metrowerks Java, or Sun's applet viewer can be used to run your Java code directly from CodeWarrior.

Finally, there is debugging support, which means "VM's that our debugger can talk to". Our long term plan is for our debugger to support any VM that implements Sun's debug API, and right now, that means the Metrowerks VM and Apple's MRJ. Obviously, the more VMs we support, the more solid your VM testing can be.We understand how important that is. Bottom line, we need to be VM-neutral, but we do play better with some VM's than others.

Dave: Can you talk about the direction of the CW VM?

Scott: I guess the best way to describe the VM's direction is that we don't intend the Metrowerks VM to be the reference standard for Java on the MacOS — it's up to Apple to do that as it moves Java into the OS. But until that day happens, we'll do everything we can to provide developers with the best tools for Java development. That means making sure that our VM keeps up with enhancements defined by Sun, and providing the additional tools that come along with those VM enhancements. To that end, we've updated the VM, compiler, and Metrowerks Java to version 1.1.x, which should have shipped in final form by the time you read this. (Sun seems to put out .x versions every other week, so who knows where we'll be when this article prints!)

In addition to the 1.1 features you would expect, the 1.1 VM will support a number of APIs that can be used to embed the VM into a developer's application. Apple's JManager API is supported, as well as Sun's JNI and Microsoft's COM.Many people have asked us if we're going to support JRI (Netscape's native interface API). When we wrote the VM for Internet Explorer, there wasn't a need for JRI, so we didn't implement it. But now that Sun's JNI is becoming the standard for a native interface to the VM, and the fact that JNI support is basically cross platform, we decided that we will support JNI in the 1.1 VM. I expect that Apple will move from JRI to JNI as well given that JRI is generally going away, so it looks like our decision was a good one.

We've also added some new features to Metrowerks Java to make it more useful as a Java development tool — things like a basic JAR file maker, being able to print the console window, more flexible tool dialogs, and the ability to choose which VM you want to work with. As I write this, we're still sorting out what will or won't be included, so there may be more than that, but I can say these will be there.

Dave: What are you doing to improve JIT performance?

Tim: Enhancement of our JIT technology is bringing an enormous speed boost to our VM's performance. We have engineers that are dedicated to optimizing our core JIT technology, and this is really having tangible effects. Between CW11 and CWPro1, our benchmark scores doubled. We expect this trend to continue, as we add more and more optimzations to our JIT.

Kevin: With respect to the JIT, there are two types of optimizations which can be done. The first concerns improvements in code quality, i.e, making the JIT compiler produce code which is more optimal in some way. The second is concerned with making the translator (JIT) itself faster or more space efficient.

Our recent efforts have been towards improving code quality, sometimes at the expense of translator speed. As Tim mentioned, we've seen a tremendous improvement in some of our benchmark scores. This is due to an optimization step which we've added which performs transformations on the internal representation of a method. One example of the type of optimization that we've done is to look for patterns which represent the logical not operator in java. Since there is no java bytecode which represents the logical not, a java compiler is forced to compile something like the following java statement:

```
cond = !cond;
```

into the following java bytecodes:

```
0 iload_0
1 ifeq 8
4 iconst_0
5 goto 9
8 iconst_1
9 istore_0
```

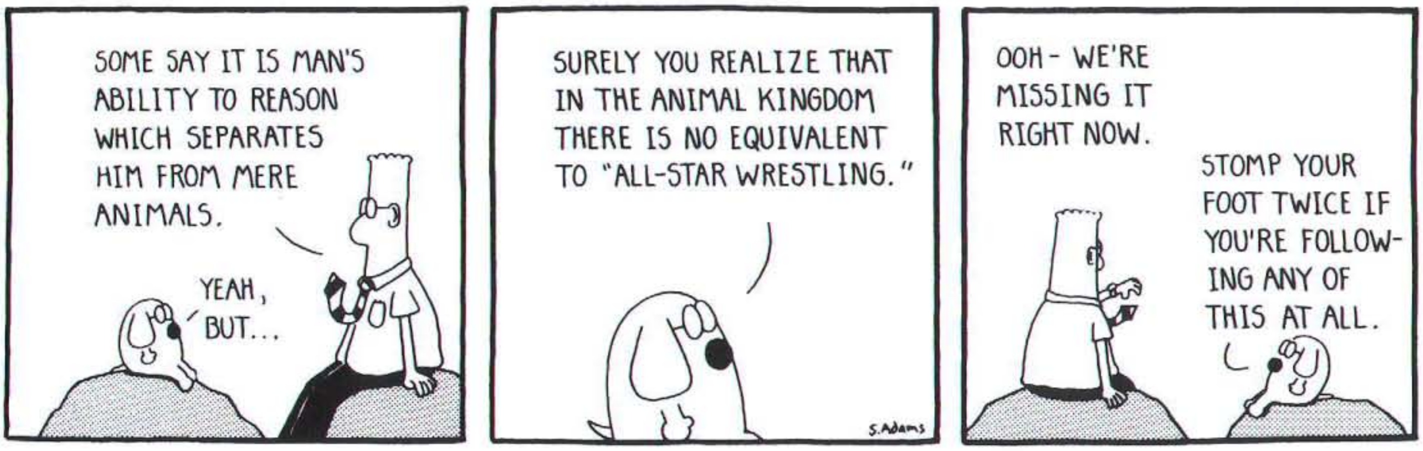
Note that there are two branch instructions. If we were to naively transform this to native code, we'd also end up with a test and two branch instructions. But on most processors we can do considerably better. What we did is to invent a new instruction called "lnot". Our optimizer recognizes patterns like the above and transforms these patterns to our new "lnot" instruction. Remember that this new instruction is a new instruction in our internal form, not some new bytecode we've added. But if we were to code it as bytecodes, the above would look something like this:

```
0 iload_0
1 lnot
2 istore_0
```

Of course there are many other optimizations that we do, but most of them are similar to the one described above where we look for certain types of patterns and then perform the obvious transformation. Our future efforts will be to do more of the same and also some tuning of the JIT compiler itself so that it will do its translation in a shorter amount of time. MT

Want to suggest an article for the magazine? Send your suggestion to <mailto:editorial@mactech.com>

Dilbert® *by Scott Adams*


SOME SAY IT IS MAN'S ABILITY TO REASON WHICH SEPARATES HIM FROM MERE ANIMALS.
YEAH, BUT...
SURELY YOU REALIZE THAT IN THE ANIMAL KINGDOM THERE IS NO EQUIVALENT TO "ALL-STAR WRESTLING."
S.Adams
OOH - WE'RE MISSING IT RIGHT NOW.
STOMP YOUR FOOT TWICE IF YOU'RE FOLLOWING ANY OF THIS AT ALL.

by Nicholas C. "nick.c" DeMello <online@mactech.com>

No coffee jokes. Simply put: it's time to get serious about Java. The promise of Java was that Java compiled code would not be restricted to a specific type of computer. While a C program must be compiled for a single chipset (Power PC, Mac 68k, Pentium, NeXT, etc.), Java was intended to be compiled for a virtual machine. Java virtual machines (Java interpreters) could be created for all platforms, so the program you compile on your Mac could then be copied to — and run on — any computer your customer might own.

That was the theory. But before it could happen Mac OS resident Java developers needed a Mac OS virtual machine. We also needed Mac OS Java compilers, sufficient documentation, tutorials and example code to get started. Guess what? The tools are online. Java is ready for us and it's time to get to work.

Starting Points

Sun has put a wealth of Java documentation online, including a description of the history and nature of Java by Jason English — definitely worth reading. A good starting point for exploring Mac OS Java resources is the Apple Flavored Java pages and make sure to check out the ACM Java Pages by Omar Patiño Sileceo for other valuable links.

At MacWorld this year Apple announced Macintosh Runtime Java 1.0—a Java virtual machine for the Mac. Mac OS 8 will incorporate MRJ 1.02 (available now on Apple's Java site), and Apple is expected to ship MRJ 1.5 (final) one month after Mac OS 8 is released. MRJ 1.5 includes a JIT (just in time) component but is not fully Java 1.1.1 compliant. Early tests indicate MRJ 1.5 will have tremendous performance boost due to its JIT component. Webintosh's JVM Showdown has an excellent discussion of what JIT is and how it effects JVM performance.

Sun's Java Documentation
<http://www.javasoft.com/docs/>
Apple Flavored Java
<http://www.mbmdesigns.com/macjava/>
The ACM Java Pages, by Omar Patiño Siliceo
<http://acm.org/~ops/java.html>
MRJ, Runtime Java for the Mac
<http://applejava.apple.com/>
The JVM Showdown
<http://www.webintosh.com/features/jvm.html>

Getting Educated

Mary Campione and Kathy Walrath have authored the definitive Java Tutorial, an amazingly deep, multi-layered document that is really five tutorials in one. The five sections offer increasingly challenging content: the basics of Java programs; writing applets; creating a user interface; networking & security issues; and integrating native methods into Java programs. Another good introductory tutorial is Elliote Rusty Harold's "Brewing Java" site.

When you get past the basics check out two advanced tutorials by Stefan Koch and Nelson Yu. Stefan's "KneeDeep" in Java site requires a working knowledge of Java, but will lead you through some very kool projects. The AWT tutorial gets into the details of how to use the abstract window toolkit to manipulate many standard GUI components such as buttons, lists, menus, and text areas. It also requires a working knowledge of Java.

The Java Tutorial, by Mary Campione and Kathy Walrath
<http://java.sun.com/docs/books/tutorial/>
<http://java.cs.uni-magdeburg.de/java/dokus/Tutorial/>
Brewing Java by Elliotte Rusty Harold
<http://sunsite.unc.edu/javafaq/javatutorial.html>
KneeDeep in Java by Stefan Koch
<http://rummelplatz.uni-mannheim.de/~skoch/javatut/kneedeep.htm>
Nelson Yu's AWT Tutorial
<http://java.cs.uni-magdeburg.de/dokus/AWT.Tutorial/AWT.Tutorial.html>

The Java FAQ at digital focus is host to two essential Java references. Click on the "How do I...?" or "Java resources" links to post and answer Java questions or tour one of the most extensive collection of Java links on the web. Usenet's comp.lang.java newsgroup has recently been divided into eight component newsgroups, but the comp.lang.java FAQ lives on. It is regularly posted to comp.lang.java.programmer, and can also be accessed on the sunsite web server. Looking for example code? Visit the Java Class Warehouse, GeoJava Corner, and Gamelin's Java Tools page.

The Java Developer FAQ
<http://www.digitalfocus.com/digitalfocus/faq/index.html>
Ho do I ...?
<http://www.digitalfocus.com/digitalfocus/faq/howdoi.html>
The comp.lang.java newgroups
<http://www.javasoft.com/aboutJava/newsgroups.html>
The comp.lang.java FAQ
<http://sunsite.unc.edu/javafaq/javafaq.html>
The Java Class Warehouse
<http://www.jtauber.com/java/jcw>
GeoJava Corner
<http://www.ggrweb.com/geojava/>
Gamelin's Java Tools Page
<http://java.developer.com/pages/Gamelan.util.html>

Parting Shots

That's it for our tour of Java. We'll leave you with a few final links to explore on your own. Keep your eye on Marimba. The Castanet technology is still rough, but very promising...

Marimba, home of Castanet, Streaming Java,
Castanet will be included in Mac OS 8
<http://www.marimba.com/>
WWDC '97, Internet Technologies Track, Where Apple is going with Java
<http://devworld.apple.com/dev/WWDC/internettechnologies.html>

by Steve Sisak

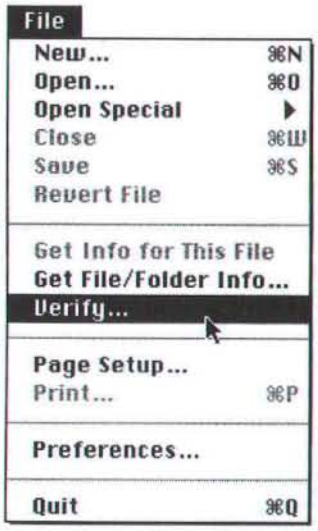
PROBLEM

Your Mac is crashing, and all your disk utilities report no problems. Yet you know something is wrong!

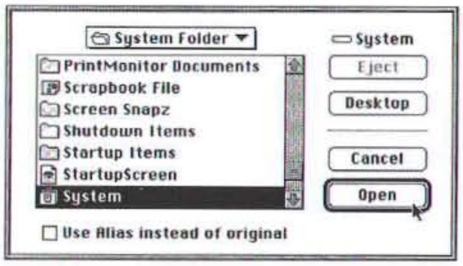
SOLUTION

Check, and if necessary, repair your System file using ResEdit. Here's how.

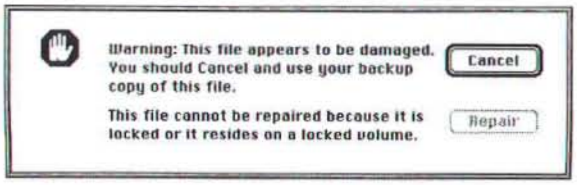
1. Start ResEdit, and select Verify... from the File menu.



2. Open your System file:



3. If ResEdit finds a problem, it will tell you:


Warning: This file appears to be damaged. You should Cancel and use your backup copy of this file.

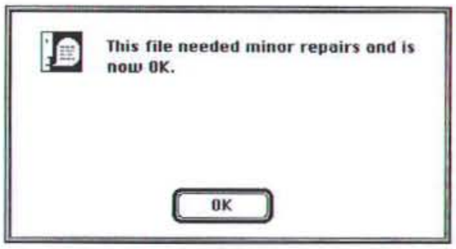
This file cannot be repaired because it is locked or it resides on a locked volume.

4. Since the file is locked, it cannot be repaired. Make a copy of the System file by

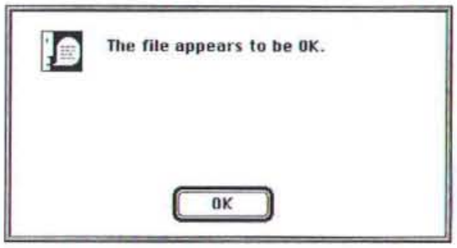
i) Press the Option key
ii) Click on the System file
iii) While holding the Option key down, drag the System file to the desktop. This will make a copy of the System file.

5. Repeat steps 1 & 2 above, but instead of selecting the System file in the System folder, select the System file that you just copied to the Desktop.

6. This time, because the file is not use, ResEdit can make the repairs.


This file needed minor repairs and is now OK.

7. Verify it one more time just to make sure.


The file appears to be OK.

8. Drag the damaged System file from the System folder and place it in the Trash.

9. Drag the repaired System file from the Desktop to the System folder.

10. Restart your Mac. All is well.

Bill Modesitt
bill@mauisoftware.com

by Jessica Courtney

Iconovex Corp. Introduces EchoSearch Web Software for the Macintosh — The First Truly Intelligent Desktop Research Tool

Iconovex Corporation, makers of the desktop-based, professional research tool EchoSearch, announced a Macintosh version, EchoSearch 1.08. A simple, yet powerful tool, EchoSearch significantly improves the efficiency and accuracy of Internet or corporate Intranet searches.

Making use of Java's multi-threading capabilities, EchoSearch enables users to simultaneously query multiple search engines. Unlike other Web search tools, EchoSearch features an intelligent back-end that can index and accurately summarize the research documents that are gathered in response to the query.

Once the complete hit list has been built, EchoSearch again uses Java's multi-threading capabilities to simultaneously download the documents and to process them linguistically, extracting the important phrases and ideas.

The resulting indexes and summaries are then matched against the user's original query and the user is presented with a condensed view that contains only the most relevant entries. These entries are then hyperlinked to their occurrences in the original documents. After viewing the research results, a user has the option of refining the original search by returning to the EchoSearch interface and clicking on the "Refine Results" button. When this alternative is selected, the program then displays another subset view of the refined query on the initial global views (documents/hits that were found the first time).

EchoSearch for the Macintosh can be downloaded for free from the Iconovex Web site at <http://www.iconovex.com>.

Lasso 2.0 Sets a New Standard for Web Database Publishing on the Mac OS

Blue World Communications, Inc. began shipping Lasso 2.0, an upgrade to Lasso, for Web-enabled database publishing with FileMaker Pro 3.0 for Mac OS. Lasso 2.0 is the only complete solution for web-enabling and Java-enabling FileMaker Pro, the leading Macintosh database from Claris Corporation. Lasso 2.0 is a fully integrated publishing solution that is well suited to any size enterprise.

Lasso 2.0 is packed with features that previously required scores of separate solutions costing thousands of dollars. Beyond advanced FileMaker Pro Web integration, Lasso 2.0 provides all the advanced dynamic Web database publishing capabilities one might expect. This includes built-in image conversion, logging, comprehensive conditionals, multi-tier remotely controlled security, ability to send Apple Events, complex Server Side Include functions and more. Noteworthy is Lasso 2.0's new "Inline" command which allows multiple database operations to be processed all within a single HTML file. Also, Lasso 2.0 can now operate as an action based upon unique file suffixes and reference format files stored anywhere across one's local network.

Unlike competitive solutions, Lasso 2.0 provides full support for FileMaker Pro features such as repeating fields and related records via portals. What's more, Lasso was the first and remains the only FileMaker Pro Web-enabling solution that contains built-in email forwarding capabilities.

Lasso 2.0 operates on the principle of easily editable HTML files which contain all database integration commands and dynamic page generation instructions alongside standard HTML page elements. This extended HTML mark-up "language" is referred to as the Lasso Dynamic Markup Language (LDML). The LDML method has proven itself superior to users who quickly outgrow cumbersome and limited "wizard-like" approaches to database integration.

Lasso users operate in their HTML authoring environment of choice and use the included GUI-based FM Link tool to drag-and-drop LDML commands into their Web pages, providing the most direct way to add database functionality while working in a graphical HTML editing environment.

Commerce Capabilities

Lasso 2.0 expands upon Lasso 1.2.1's already rich commerce capabilities providing full cookie support and the ability to pass tokens along with the ability to send Apple Events to MacAuthorize for credit card verification. With Lasso's built-in email forwarding capability, one sends customer order receipts immediately via email. Sample online shopping cart templates and databases are provided which assist users with rapid cyberstore development.

Java-Enabling FileMaker Pro

Lasso 2.0 allows Java applets and applications to access FileMaker Pro databases from anywhere on the Internet. Java developers use the provided set of Java classes to create dynamic front-ends to FileMaker Pro data. Sample applets provided by Blue World include a Java-based FileMaker Pro client that mimics the FileMaker Pro standard interface, a Java-based chat forum code-named "CoffeeHouse", a Lasso Tags database and a database info gathering example.

Built-in Web Server

Lasso 2.0 Server version offers a built-in Web server optimized for dedicated FileMaker Pro Web database servers. Beyond all the capabilities found in the CGI and plug-in versions, Lasso 2.0 Server provides standard Web server features including: the ability to serve HTML, GIF, and JPEG files (in addition to other MIME types), selectable port controls, suffix mapping, process controls based upon suffix name and more.

Free Solutions

To assist users with building dynamic Web database solutions, Blue World provides numerous free templates and databases. Solutions include online shopping carts, guestbooks, discussion forums, Web calendars, numerous Java examples and more. A listing of solutions is available at <http://www.blueworld.com/lasso/solutions/>.

Further information about Lasso including links to online examples and free run-limited versions is available on the Lasso website at <http://www.blueworld.com/lasso/>.

StuffIt InstallerMaker 4.1 with Electronic Commerce Capabilities now Available from Aladdin Systems, Inc.

Aladdin's Installer Gets New Tasks and Flag Alerts for Improved Installations

Aladdin Systems, Inc., developer and publisher of the worldwide compression standard StuffIt, today announced StuffIt InstallerMaker 4.1 began shipping. StuffIt InstallerMaker allows developers to easily create custom software installers and automatic trialware software (a time-locked demo) without any coding. InstallerMaker is the only Macintosh installer with automatic trialware creation capability. Trialware created with InstallerMaker can be purchased by end users with the click of a button.

Six New Features in InstallerMaker 4.1

Two new Tasks make InstallerMaker faster and easier to use. The "Display Alert" Task allows developers to show alert messages to users at any time during installation. The "Launch" Task allows developers to create tasks to launch applications, open other files, or launch additional installers that are necessary to the installation of their software, without having to include them in the installer archive.

"Preserve Success/Fail Status," and the "Execute When" features let users know when and if their installation succeeded. Preserve Success/Fail Status, allows developers to preserve the flag that indicates whether the last Task and/or Find failed when executing tasks, and the Execute When condition. Execute When is a new feature that allows Tasks to be set to execute under three conditions: Always, If last Task/Find succeeded, or If last Task/Find failed.

The fifth new feature is the "Edit Installer Finder Info." It allows developers to set the Shared and File Locked flags for the installer, and/or the minimum and preferred memory allocation. This feature makes it easier to retain the custom settings for the installer.

The sixth new feature included in InstallerMaker 4.1 is added support for calling IMid extensions before and after Alias and Rename tasks.

InstallerMaker 4.1 features time saving enhancements including: the capability to use ShrinkWrap 2.0 and Aladdin's upcoming ShrinkWrap 3.0 for creating disk images as part of the build process, a software license display mode during installation, a new update architecture that creates software updaters in one tenth the time and up to 60% smaller, support for building additional foreign language installers, a faster find, easier

Internet TCP/IP configuration for ISPs, the ability to display banners and play sounds during the installation process, as well as valuable user interface enhancements.

Electronic Commerce Solution

With InstallerMaker 4.1, developers can convert their software into trialware with two simple menu selections; setting the purchase price and demo period, and adding graphics. InstallerMaker creates a version of trialware with full featured electronic commerce, a fast, simple extension of the installation creation process which developers are already required to do, but without the extensive additional coding necessary on other electronic commerce solutions.

When InstallerMaker1s trialware feature is enabled, the user is notified every time they run the software of the number of days or launches left in their trial period. They are also given the opportunity to purchase the product from within the application. The purchase may be made in real time via the Internet, modem connection, an 800 phone call, by fax, or postal mail. When the demo period expires users will not be able to continue using the software without a purchase even if they reinstall it.

Creating Disk Images with ShrinkWrap

InstallerMaker has the capability to use ShrinkWrap to create disk images while building an installer. This eliminates a time consuming step in the SCM process and results in more reliable product builds. With Aladdin's ShrinkWrap technology InstallerMaker can now produce the master disk images for duplication automatically.

Software License Dialog With Foreign Language Support

InstallerMaker supports a separate software license dialog box during installation. Developers can add copies of their software license agreement in multiple languages (Spanish, French, German, etc.) to be displayed before installation. The end user must accept the terms of the license agreement before the installation will proceed.

Internet Configuration Support

New documentation and the Internet PrefSaver have been added to assist developers, in configuring a Mac OS computer for Internet access. PrefSaver helps ensure user account name and password information is propagated across common Internet software packages such as Netscape Navigator, Eudora Lite, Microsoft Internet Explorer, Internet Config and many others through Internet Config.

StuffIt InstallerMaker is distributed electronically through the World Wide Web at <http://www.aladdinsys.com> and on CompuServe. InstallerMaker is sold through license agreements directly from Aladdin. Current licensees can download a free upgrade of the new version.

Stalker Software adds Anti-Spamming Features to its Mail Server Products

Stalker Software, Inc. has just released its latest feature, Anti-Spamming, for both the CommuniGate Messaging System and the Stalker Internet Mail Server. The new feature is especially popular for Internet Service Providers as they suffer the most from senders of bulk email (spammers). Not only do Spam offenders fill mailboxes that ISPs maintain for their clients, but they also abuse ISP mail servers, using them as relays to distribute bulk email all over the world.

Anti-spam features are included into the both Stalker SMTP products: the commercial CommuniGate SMTP module, and into the freeware Stalker Internet Mail Server (SIMS).

The first feature allows the server administrator to put the network addresses of spammer hosts into the server Black List: all connections from those hosts are rejected, and no mail is taken from those hosts. There are already several web sites on the Internet that maintain a list of offending hosts addresses: information from those sites can be used to fill the Black List.

The second feature allows the administrator to restrict relaying features of SMTP servers, thus making the servers useless for spammers. The server administrator can specify the network addresses of all "legitimate" clients of that server. Usually, it's just all the addresses on the server LAN, but it can be any set of addresses.

While messages submitted from those "listed" addresses can be sent (relayed) to any address on the Internet, and messages received from any Internet address can be sent to any "listed" system, messages from "unlisted" addresses are not relayed to other "unlisted" addresses. This makes the host useless for spammers that would want to employ it for relaying their messages all over the Net.

These new versions of the CommuniGate SMTP and the Stalker Internet Mail Server implement the standard mail protocols (POP,SMTP,POP,PWD) via AppleTalk, in addition to TCP/IP. Now mailers that support AppleTalk connections, such as the Claris Emailer, can use both Stalker standard-based servers on AppleTalk-only LANs — the feature that was available only with proprietary products such as Claris OfficeMail.

These latest updates allow Stalker to offer the special package of (25-users license with UUCP, POP, SMTP, and PWD modules, regular price $400) for just $189 to all owners of the Claris Office servers.

The Stalker Internet Mail Server can be downloaded from <http://www.stalker.com/SIMS>. This product is free, but it is backed up with the Stalker technical support staff and has a dedicated mailing list.

To allow all Macintosh mail server administrators to employ these new features as soon as possible, a utility to convert Apple/Eudora Mail Server accounts into Stalker Internet Mail Server accounts is posted on the Stalker Web site at <http://www.stalker.com/SIMS/Converter.html>.

The CommuniGate SMTP and other components of the CommuniGate Integrated Messaging System can be downloaded from <http://www.stalker.com/CommuniGate>.

Updates to the new version of the CommuniGate SMTP module are free.

ClearWay Technologies Ships FireSite Web Site Accelerator 2.1 Optimized for ISDN-based Web Servers

Boston, MA., July 11, 1997 — ClearWay Technologies, Inc. today announced an updated version of its FireSite(tm) Web Site Accelerator, which contains special optimizations for ISDN-based Web servers. When used with ISDN lines running at speeds from 56Kbps to 128Kbps, FireSite 2.1 can sustain page delivery speeds exceeding 500Kbps, and can increase Web site "traffic capacity" by 400-800%. FireSite 2.1 also can now operated in conjunction with "virtual ISDN connections". Free demos are available from <http://www.clearway.com/pages/demos.html>.

Many ISDN Webmasters are now hosting additional "virtual" Web sites, and FireSite 2.1, with its integrated Virtual Domain Manager, provides an excellent solution. FireSite 2.1 Standard and Multimedia Edition include the FireSite Virtual Domain Manager, the Web's most advanced single-IP Web hosting system. Because server load and bandwidth requirements increase as more sites are hosted, FireSite's speed and capacity increases are a perfect companion for Webmasters hosting virtual Web sites on ISDN lines.

The list of all FireSite 2.1 products and prices is available at <http://www.clearway.com/pages/orderpage.html>.

Apple Computer Bundles EveryWare's Tango Enterprise and Butler SQL

EveryWare Development, Inc. reached a software distribution agreement with Apple Computer, Inc. The agreement is to bundle Tango Enterprise and Butler SQL, EveryWare's intranet rapid application development tool and relational database management system with the Apple Internet Server Solution.

Apple Computer, Inc. is focused on providing a complete solution to organizations in business, education, or publishing that want to establish a significant presence on the Web. The Apple Internet Server Solution, which is one of three available software solutions for Apple's Workgroup Server family, addresses the unique demands of the Web server market by combining software from industry-leading Internet developers with Apple's powerful new Workgroup servers.

Tango Enterprise is a leading-edge development tool for creating dynamic, Web-based applications for the Internet or corporate intranets. Tango's visual development environment allows Web developers to rapidly develop and deploy sophisticated Web applications that are integrated with databases. Tango Enterprise will integrate two of the most popular databases on the Macintosh: FileMaker Pro from Claris Corp. and Butler SQL from EveryWare Development Inc. Tango Enterprise will also work with all other ODBC compliant databases including Oracle, Informix, SQL Server, Sybase, Access, and DB2

The Tango product family allows for rapid construction of dynamic Web sites. The family consists of Tango Enterprise, Tango for Access, Tango for FileMaker, and Tango Merchant. The Bolero product family enables Web administrators to effective analyze and maintain high volume Web sites. This product is presently shipping on the Macintosh. MT psi Scientific Systems

PERCEPTIVE SCIENTIFIC INSTRUMENTS, INC.

Macintosh Developers for US & UK

Perceptive Scientific Instruments, Inc. is a global leader in the development and marketing of digital imaging systems for genetics applications. The company has offices in the United States and the United Kingdom.

We are currently looking for Macintosh Software Developers for both our US and UK locations to work on our next generation products for the genetics market place. Candidates should have 2-5 years of Macintosh programming experience in C/C++ with strong analytical skills. Experience with object oriented programming methodologies will be a plus.

Please send your resumé to H. Baxi, PSII, 2525 S. Shore Blvd., League City, TX 77573, USA or fax to +1 (281) 538-2222 or email to hbaxi@persci.com.

http://www.scientific.com

Professional software developers looking for career opportunities should check out our web site. We offer nationwide employment assistance, resume help, marketability assessment, and never a fee to the applicant. 800-231-5920, Fax 800-757-9003 eMail:das@scientific.com

Scientific Placement, Inc.

Visit MacTech® Magazine's Web site!
http://www.mactech.com

Telecommuting job opportunity for a programmer experienced with cross platform development in C/C++. Experience writing plug-ins for Adobe Acrobat a plus. This is an employee position with benefits, but without the hassles of commuting to a regular office. Potential candidates must be focused and able to work independantly. Send resumes (in PDF format) to vgavin@digapp.com.

Digital Applications, Inc.
215 East Providence Road
Aldan, PA 19018-4129
http://www.digapp.com

Live and Work in Japan

Individuals with programming and teaching skills needed to work in Tokyo on a one year contract. Please fax or e-mail your detailed resume to: **International Education Services** fax 81-3-3498-7113 or e-mail to iesinfo@iac.co.jp Interviews with qualified applicants will be held in September and October this year in various locations in the United States.

The Law Office of Bradley M. Sniderman

California Lawyer focusing on Intellectual Property, Corporate, Commercial and Contract law, as well as Wills and Trusts.

If you are looking to protect your software with Copyright or Trademark protection, or if you need help establishing or maintaining your business, please give me a call or an e-mail. Reasonable fees.

(310) 553-4054
Brad@Sniderman.com

MacTech® Magazine is your recruitment vehicle

When you need to fill important positions at your company, MacTech® Magazine is the consistent choice of companies across the country for hiring the best qualified Macintosh programmers and developers. Let MacTech® Magazine deliver your recruit-ment message to an audience of over 27,000 qualified computer professionals.

Call Ruth Subrin at 805/494-9797

ADVERTISER/ PRODUCT LIST

LIST OF ADVERTISERS

Aladdin Knowledge Systems Ltd.	5
Aladdin Systems, Inc.	15
Bare Bones Software, Inc.	7
BeeHive Technologies, Inc.	32
Bowers Development	49
Datapak Software Inc.	11
Developer Depot™	20, 87
Digital Applications, Inc.	68
Excel Software	33
Faircom Corporation	16
International Education Services	68
Internet Source Book	37
MacTech® CD ROM	29
MacTech® Now!	17
Mango Tree Software, Inc.	57
Mathemaesthetics, Inc.	1
Metrowerks	BC
Micro Macro Technologies, Ltd.	9
Neologic Systems	21
Onyx Technology	25
Perceptive Scientific Instruments	68
PrimeTime Freeware	19
Purity Software	IBC
QKS	35
Ray Sauers Associates	39
Rivers Edge Corp.	51
Scientific Placement	68
Seapine Software, Inc.	53
Snowbound Software Corp.	45
Symantec	IFC
Water's Edge Software	26

LIST OF PRODUCTS

ADB I/O • Beehive Technologies, Inc.	32
AppMaker • Bowers Development	49
BBEdit • Bare Bones Software, Inc.	7
CodeWarrior™ • Metrowerks	BC
c-tree™ • Faircom Corporation	16
Developer Tools • Developer Depot	20, 87
DragInstall • Ray Sauers Associates	39
FairCom® Server • Purity Software	IBC
Heap Manager • BeeHive Technologies Inc.	32
InstallerMaker • Aladdin Systems, Inc.	15
Internet Source Book • Internet Source Book	37
MacA&D • Excel Software	33
MacHASP • Aladdin Knowledge Systems Ltd.	5
MicroGuard Plus™ • Micro Macro Technologies, Ltd.	9
MK Linux • PrimeTime Freeware	19
neoAccess™ • Neologic Systems	21
ODBC Driver • Faircom Corporation	16
OpenPaige™ • DataPak Software, Inc.	11
RasterMaster 6.0 • Snowbound Software Corporation	45
Recruitment • Digital Applications, Inc.	68
Recruitment • International Education Services.	68
Recruitment • Perceptive Scientific Instruments	68
Recruitment • Scientific Placement	68
Resorcerer® 1.2 • Mathemaesthetics, Inc.	1
RiverReader-Laser • Rivers Edge Corporation	51
r-tree® • Faircom Corporation	16
SmalltalkAgents® • QKS	35
Spotlight™ • Onyx Technology	25
TestTrack™ • Seapine Software, Inc.	53
TCP/IP Scripting Addition • Mango Tree Software	57
Tools Plus™ • Water's Edge Software	26
Visual Cafe™ • Symantec	IFC
WebSentinel™ • Purity Software	IBC
WebSiphon™ • Purity Software	IBC

The index on this page is provided as a service to our readers. The publisher does not assume any liability for errors or omissions.

by Éric Simenel

My first Yellow Application

Since most of us are in the process of exploring the OpenStep (Yellow Box) framework and its development environment, it's likely that we are going to encounter the similar pitfalls. I thought it would be useful to describe my first experience at writing something more ambitious than a simple snippet, so that you could benefit from my trials. To make the most out of this article, you'll need to already have some knowledge of Yellow Box. This article is not a tutorial, it's more like a journal...

First, let me give you a little background. I've been programming for the past 17 years (13 on Macintosh) using a lot of languages (in order of appearance: FORTRAN, 6502 asm, BASIC, COBOL, LISP, Pascal and Object Pascal, 68K asm, C and C++, PowerPC asm and Java), and a lot of environments and Operating Systems (micros, minis and mainframes).

As far as frameworks go, I've only used a very little bit of MacApp and more of ODF (the late OpenDoc Development Framework), but mainly I've used the Mac Toolbox when developing company and personal stuff. The main reason is that I've been developing with it for so long that it's just like an old friend; you know what you can ask it to do and what you'd better not.

Since I've been doing Object Oriented Programming for the last 6 years, I've always regretted a little bit that the Mac Toolbox doesn't have OO APIs. Of course, I could have used a framework but this adds an extra layer which you sometimes appreciate and sometimes don't.

Well, with Rhapsody, it's OOP from the word go, so no more excuses.

At first, the Objective-C syntax with all of its "["s and "]"s looks really different, but I'd say that after a day or two, you don't even think about it anymore. And anyhow, the Developer Release or the Premier Release will come with the Objective-C modern syntax which will be more familiar to the C/C++ developer.

Although a lot of people admire Interface Builder (IB), and it's indeed a very nice tool, I, for one, think that the real power comes from the Yellow Box framework. When I developed my first Yellow application, I spent maybe 5-10% of my time in IB and the rest in Project Builder (PB). I must add that if IB didn't exist, we would need to spend an incredible amount of time writing source code to do what it does since this tool is much more than a simple graphic user interface designer tool, but more on that below...

While writing my sample code, I learned quite a few things which I didn't find explicitly in the documentation, so I thought it might be useful to share them. I must add that I later found some (but not all) of this information in documents which I didn't possess at first, and since I believe this will be the case for most of you, and that, anyway, who reads the entire documentation before starting coding, it's still useful to have those points collected in a useful way, i.e. this article.

I guess readers of the late develop magazine will remember that I collect Comics Books, Comics in short, and my non-trivial sample code is the port of an existing Comics Database Management Macintosh application to the Yellow Box. There is an additional document "The way I collect Comics" in the development folder, which explains why I designed the user interface the way it is. This article deals with the "how" I designed it.

The development folder containing all the source code, interface files, project files, etc. is on the Rhapsody Developer Release CD, and will also be on our web site, whose address is not known at this date but which I trust you shouldn't have too much difficulty to find.

This article contains extracts of the source code only when relevant, you might find it useful to read the entire source code along with this article.

Éric Simenel is really happy he transferred in Cupertino's DTS from Paris' DTS. Aside from the fact that he got a real good welcome from his current colleagues, he's getting much more sun here than there, and, due to his constant location here, he has easier access to Comic Books Conventions where he completed many runs... The current mark is at 22,000 and counting.

What's in a Nib (NeXT Interface Builder) File

Everything which has been created in the nib while using IB (i.e., every icon which you see in the "Instances" panel), will be instantiated when the nib opens.

That means that when your application launches, its main nib is opened, all objects which have an icon in the "Instances" panel will be instantiated, then their init method will be called, then the outlet connections you have established with IB will be filled with the appropriate value, and then their awakeFromNib method will be called.

As an example, since my Yellow application deals with one unique Comics database object, instead of allocating it in the source code somewhere, I just instantiated it in the nib file. The nib file is not just for graphic user interface elements.

This is also one of the reasons why you should use subprojects with their own nib files. You load these nib files with the loadNibNamed: method (from NSBundle) when needed which saves on memory usage and improves performance. And speaking of memory...

Memory Problems

Always when, and **only when**, you **explicitly** call the alloc class method, or the retain, copy or mutableCopy instance methods, you must release or autorelease the object thus allocated. All other objects you get through **another** method, you **neither** release nor autorelease .

That means that if you write

```
    NSAClass *obj = [[NSAClass alloc] init];
```

then you must also do either [obj release] if the method where you allocated it is the only one using it, or [obj autorelease] if the method is returning this object to somebody else. But if you write, for example.

```
NSFileManager *nsfm = [NSFileManager defaultManager];
NSData *data = [nsfm contents AtPath:fileName];
```

then don't write [nsfm release]; nor [data release]; or you'll end up in trouble (I did).
Also, if you write

```
NSString *str = [[NSString alloc] initWithCString:myCString];
```

then you must write [str release]; or [str autorelease]; depending on the usage, but if you write

```
NSString *str = [NSString stringWithCString:myCString];
```

then don't release str, (in this case it has already been autoreleased by the class method stringWithCString:).

Memory Protection

Although when you mismanage memory in the Yellow Box, it's not as bad (directly) as on the Mac OS, you may still have some surprises. These problems come from releasing an object too many times (see above), going outside the range of an NSArray object, and certainly other ways which haven't bitten me yet. In all those cases of memory mismanagement, if you launched your application from the launcher/debugger window in PB, you'll get explicit error or exception messages. You then fix the problem in your source code, build and test again. Afterwards, in some cases, although everything looks fine, you'll experience minor (but irritating) annoyances: PB won't build anymore (the compiler keeps sending obscure messages), PB won't index your project(s) anymore, IB won't save your modifications, or PB or IB will quit suddenly and then refuse to launch again.

Depending on the exact symptom, I found the following solutions: quit PB and IB, relaunch them, everything's fine (this works 50% of the time), if not, log out, log in again (takes just a couple of seconds), relaunch them, everything's fine (this works 45% of the time), if not, power off, then cold start again (this works 5% of the time). In any case, I never lost any data.

On a brighter side, the more I developed with this framework, the less I wrote these silly memory mismanagement mistakes, meaning that I have less and less trouble with the tools themselves. And anyway, those bugs are currently being tracked and will be fixed.

MORE ABOUT MEMORY

All objects in the Yellow Box are refcounted. The alloc class method allocates the object and sets its refcount to 1. The retain instance method increments the refcount, the release instance method decrements the refcount and if it becomes 0, deallocates the object. The autorelease instance method places the object in an autorelease pool. When the current event ends, or in other words, when the control returns to the Main Event Loop, all objects in this pool will be sent the release message once, and if their refcount becomes 0, they are deallocated. The copy instance method will do the same thing as the retain method if the object is immutable. If the object is mutable, copy will create a new object with the same content and a brand new refcount of 1, just as mutableCopy does.

You should never call the dealloc method yourself.

Whoever creates an object **or increments** its refcount, by any of the provided means, **is responsible** for decrementing its refcount by calling release or autorelease . That's the reason why autorelease is provided. For example, when you create and return an object to your caller, the caller has no idea how you created the object; it could be an allocation, but it could also be a static or on a stack or whatever. Therefore the caller must **not** release this object, **you** have to. But your problem is that you may no longer have access to this object later when the moment comes. What you do, in this case, is to autorelease the object before you return it. If the caller wishes to keep it, then the caller must do a retain on it, but if the caller doesn't care, then the object will be automatically released at the end of the current event since nobody retained it.

Updating Window Content

The updating mechanism is very different from what you are used to on Mac OS. By default your window is buffered (as in offscreen), it can also be retained and non-retained, but I must say I was rather surprised by the standard graphic objects reaction to those choices, try it and you'll see what I mean. It looks like some of these objects assume they are living in a buffered window and have their own bit cache management relying on that assumption, and this goes awry whenever the window is not buffered. All the standard graphic objects call you (as a developer) to be able to redraw, such as drawRect: (for an NSView) or they will ask for some data as in willDisplayCell:atRow:column: (for NSBrowser) and then draw.

So far, so good; it's not really different from the Mac OS frameworks we know. The difference comes from the fact that you will not be called at all most of the time. If your window was in the background and comes to the foreground, chances are that the update will occur from the offscreen buffer and your redraw method won't be called at all (so if you put in some debugging display like I did at first, don't be surprised if you don't see it).

Some Slightly Annoying Surprises

Although the APIs of NSBrowser and NSTableView are very similar, their mechanism is quite different. To use NSTableView, you must (among other things) implement the numberOfRowsInTableView: method and, as its name suggests, you must return the number of rows of the table. To use NSBrowser, you must (among other things) implement the numberOfRowsInColumn: method. The difference is that numberOfRowsInTableView: is called a **zillion** times by the NSTableView object, whereas numberOfRowsInColumn: is called only **once** by the NSBrowser object whenever you click on a cell in a column which will make the browser fill a new column to its right.

So your strategy as a developer must be quite different. In a browser, it is the right thing to do the computation of the numbers of rows of the column you're being asked for in numberOfRowsInColumn:, whereas, in a table, you'd better do the intensive computation **only** at awakeFromNib (or init time, depending on your situation) time, and then only if the number of rows changes, save off this number of rows in a field of your controller so that you can just return this value when asked in numberOfRowsInTableView:.

Setting up IB and PB

The default rules for indenting text in the PB text editor are not necessarily what you'd wish they are. Don't hesitate to modify them in the "Preferences" window till the indentation is as you like. You'll need to set up the "Indentation" panel criteria **and** the "Key bindings" panel criteria.

If you happen to close a Palette, using the "Close Palette" menu item of the "Palettes" menu in IB, this action won't close the Palettes window, but remove the current selected palette icon from this window. To reinstall it, use the "Preferences" window of IB.

GDB

Since it is rather unlikely that we write perfect source code the first time, it is useful to follow what's happening in the debugger, or at least in the console. That's where gdb (gnu debugger) makes its entrance.

What happens to me the most, and following the works of my colleagues, I found that I am not the only one, is that I design my interface with IB, write my source code with PB, build and test and nothing happens. The stuff I just wrote just does not do what it is supposed to. In nearly all these cases, what's wrong is that I had forgotten to establish crucial connections with IB (such as a delegate or a datasource, or even an action method). Most likely you will experience the same problem.

So the first step is a double one: verify your connections again with IB **and** add some tracing code using NSLog (and don't forget to launch your application through the launcher/debugger window of PB, instead of launching it from the Workspace; if you do, however, you can still see your NSLog messages in the main console window which you can display from the "Tools" menu of the Workspace).

The prototype of NSLog is

(void)NSLog(format, ...);

and you use it as you would use printf, except that NSLog has an extra format "%@", which enables the printing of the description of an object (very useful).

If you end up being halted in gdb because of an exception or an error, the following useful commands (also accessible from the graphic user interface of the launcher/debugger window) will help you determine the cause of the problem. There are many more commands of course, so read the gdb documentation or use its help command to find out more.

Table 1 *Useful commands in gdb*

Command	Description
`po <object>`	prints the description of an object
`print * <object>`	prints the content of an object
`print * <structure>`	prints the content of a structure
`print expression`	prints the evaluated expression
`bt`	prints the stack crawl
`future-break <method>`	sets a breakpoint for code not loaded yet
`kill`	terminates the process being debugged

Note: You can also evaluate and call methods from known objects such as

print * [comicsBase titles] or

print * (NSScroller *)[self verticalScroller]

Last Note

Before we enter the detail of each class in my first Yellow application, I must say that I was pleasantly surprised by the speed of my development. In less than 3 weeks (and that's my **first** Yellow application, meaning that I spent a lot of time in the documentation trying to understand how everything works), I reached a level (converting data from the Mac OS database, archiving and unarchiving data the Yellow Box way, displaying the database content in 4 different ways, and modifying the data in a rather complex user interface) which had taken me 2 or 3 months to reach in C++ with the Mac Toolbox on Mac OS (remember, this is a port from a Mac OS application which I've been using for years). So, whenever you hear some guy speaking about Rapid Application Development (RAD) regarding the Yellow Box framework, that's the real untarnished truth, not just some sales talk.

One last thing before we go, the Yellow Box framework, like MacApp, ODF and certainly others, is a Model-View-Controller kind of a framework. Very classic these days, very powerful, and very helpful for object reuse. When I talk later about a View (i.e. user interface), I mean it in the Model-View-Controller context, so don't confuse it with an NSView which is a specific class of the Yellow Box.

ONWARD WITH THE DETAILS

ComicsObj

This is the Model part of my application. It only deals with data, not user interface.

The CComics class contains mainly an NSMutableArray of CTitles objects which contains mainly an NSMutableArray of CIssues objects. The important methods are initWithCoder: and encodeWithCoder: which are implemented in all 3 classes to allow the archiving and unarchiving of the database (look into the source code to see their implementation, nothing really difficult), sortArray:withBrand:withSeries:withKind:withState:withSort: (in CComics) which not only sorts but also extracts the appropriate titles depending on the criteria parameters, and which will be called from nearly all the controller objects of the user interface, and various accessor methods for each class.

Since the comics database will be accessed from nearly all the controller objects, it was expedient to set up a global object (of class CComics) named comicsBase declared in ComicsObj.h. It could also have been an outlet, paired with an accessor, in the main controller (see the above paragraph about nib files), and then accessed from everywhere with [[NSApp delegate] comicsBase]. This is mainly a **style** choice from the developer.

Since the comics database already existed on Mac OS and I was reluctant to reenter all the data (see the "The way I collect Comics" document for further explanation), the init method of CComics can either convert the Mac OS database or load the Yellow Box database thus:

```
#if realLoad
   self = [NSUnarchiver unarchiveObjectWithFile:fileName];
   [self retain];
   comicsBase = self;
#else
   comicsBase = self;
   [self setTitles:[NSMutableArray array]];
   [self _convertFromMac];
   [self save:nil];
#endif
return self;
```

You'll note that in the first case, comicsBase must be assigned only after the unarchiving (self is changed), whereas in the other, it must be assigned before the call to _convertFromMac which uses this global object.

Since this conversion only has to occur once, unless I destroy the application or have data corruption, I do a special build setting realLoad to 0 to convert the database. Afterwards, realLoad is set back to 1.

VerifyController (the main controller)

The Controller for the "Verify" window which is a View.

This is a very basic use of an NSTableView. Just don't forget, in IB, to connect both the dataSource and delegate outlets of the NSTableView object to this controller and implement the 2 minimum requested methods numberOfRowsInTableView: and objectValueForTableColumn:row: and your table will be fine.

To be able to return appropriate content in objectValueForTableColumn:row:, you should also not forget to give an identifier to each column in the IB inspector window; this identifier may or may not be the same as the column title (your choice, in most cases, it's a good idea to give them the same name, it makes source code writing easier later).

Since this is the main controller, it also provides the appropriate action methods (to be connected with the appropriate menu items in IB) to deal with the objects from the subprojects (browser, calendar and title longevity). Furthermore, as an NSApp delegate (don't forget to establish the connection in IB), it also implements the applicationShouldTerminate: action method which, in this case, will lead to the saving of the comics database if it has been modified.

Each time the user chooses a new choice in the popup buttons, the selChanged method is called, which returns a new array of the appropriate CTitle objects, saves the new number of rows (see the above paragraph about the number of rows in an NSTableView), and calls the private method _update which, by calling the noteNumberOfRowsChanged and reloadData, will force a redisplay of the table.

Listing 1 *Extract from VerifyController.m*

```
- (void)selChanged
{
    [comicsBase sortArray:array withBrand:brand
     withSeries:series withKind:kind withState:state
     withSort:sort];
    nbRows = [array count];
    [nbSelTitles setIntValue:nbRows];
    [self _update];
}
- (int)numberOfRowsInTableView:(NSTableView *)tableView
{
    return nbRows;
}
- (id)tableView:(NSTableView *)tv
objectValueForTableColumn:(NSTableColumn *)tableColumn
row:(int)row
{
    CTitle *thisTitle = [array objectAtIndex:row];
    NSString *identifier = [tableColumn identifier];
    if ([identifier isEqualToString:@"Abb"])
        return [thisTitle abb];
    else if ([identifier isEqualToString:@"Title"])
        return [thisTitle title];
    else return [thisTitle listIssues];
}
- (void)_update
{
    [verifyView noteNumberOfRowsChanged];
    [verifyView reloadData];
}
```

BrowserController

The Controller for the "Browser" window which is a View.

This is a very basic use of an NSBrowser. Just don't forget, in IB, to connect the delegate outlet of the NSBrowser object to this controller (the File's owner icon should be of custom class BrowserController since this is a subproject) and implement the 2 minimum requested methods numberOfRowsInColumn and willDisplayCell:atRow:column:, and your browser will be fine. Also remember to call [[browser window] makeKeyAndOrderFront:nil]; in the awakeFromNib method or else you won't see any window and lose some time understanding why.

Since the browser is defined in a subproject of the application project, we have to load its nib file, when the user wants to see this window. The menu item he will use is connected to the

```
- (void)newBrowser:(id)sender { [[BrowserController alloc]
init]; }
```

action method in the VerifyController which is the main controller. The init method of BrowserController will do the nib loading (among other things):

Listing 2 *Extract from BrowserController.m*

```
- (id)init {
  if (self = [super init])
  {
    // the browser lies in a subproject, so let's get its nib
    if (![NSBundle loadNibNamed:@"Browser"
    owner:self])
    {
      NSLog(@"Unable to load Browser.nib");
      [self release];
      return nil;
    }
    array = [[NSMutableArray alloc]
            initWithCapacity:1500];
  }
  return self;
}
- (int)browser:(NSBrowser *)sender
numberOfRowsInColumn:(int)column
{
  short brand, series, state, kind;
  // the first 4 colummns have all only 3 cells
  if (column < 4) return 3;
  // if not, let's see what the user has currently selected in the first colums
  brand = [sender selectedRowInColumn:0];
  series = [sender selectedRowInColumn:1];
  state = [sender selectedRowInColumn:2];
  kind = [sender selectedRowInColumn:3];
  // so that we can retrieve the appropriate titles (we don't care for
  // sorting in this browser)
  [comicsBase sortArray:array withBrand:brand
    withSeries:series withKind:kind withState:state
    withSort:0];
  return [array count];
}

- (void)browser:(NSBrowser *)sender
willDisplayCell:(id)cell atRow:(int)row
column:(int)column
{
  switch(column)
  {
    case 0: switch(row)
    {
      case 0: [cell setStringValue:@"All Brands"]; break;
      case 1: [cell setStringValue:@"Marvel"]; break;
      case 2: [cell setStringValue:@"DC & Others"];
              break;
    } break;
    case 1: switch(row)
    {
      case 0: [cell setStringValue:@"All Series"]; break;
      case 1: [cell setStringValue:@"Long"]; break;
      case 2: [cell setStringValue:@"Mini"]; break;
    } break;
    case 2: switch(row)
    {
      case 0: [cell setStringValue:@"All States"]; break;
      case 1: [cell setStringValue:@"Dead"]; break;
      case 2: [cell setStringValue:@"Live"]; break;
    } break;
    case 3: switch(row)
    {
      case 0: [cell setStringValue:@"All Kinds"]; break;
      case 1: [cell setStringValue:@"Main"]; break;
      case 2: [cell setStringValue:@"Dual"]; break;
    } break;
    case 4: [cell setStringValue:
              [[array objectAtIndex:row] title]];
            break;
  }
  // the 5th column is the last
  [cell setLeaf:(column == 4)];
  // this means this cell is ready for display
  [cell setLoaded: YES];
}
```

TitleLongevityController and TitleLongevityView

The Controller for the "Title Longevity" window which is a View, and the custom view inheriting from NSView which implements the graph.

This is a very basic use of an NSView (NSCustomView in IB). The TitleLongevityView is just a graph, and the controller allows the user to change the criteria of the graph (through the popup menu buttons). TitleLongevityView has to implement just 2 methods (initWithFrame: and drawRect:) to work fine. Each time a criteria changes, the display (of TitleLongevityView, inheriting from NSView) method is called, which will, at some point, calls the drawRect: method.

Never call drawRect: YOURSELF

Before drawRect: can be called, the graphic context has to be set right (the Mac OS Toolbox equivalent would be a SetPort). That's what the Window Server does, and what you should never try to attempt yourself. If you want drawRect: to be called, call display instead, and the Window Server will do the right thing for you.

The only thing you have to pay attention to is the fact that, the origin is in the lower left corner of the view (this is Display Postscript), instead of the upper left as you might be used to in QuickDraw. To draw, you can mix C-ified Postscript calls such as PSmoveto, PSlineto, PSstroke and framework calls such as drawAtPoint:withAttributes: (of NSString).

Listing 3 *Extract from TitleLongevityController.m*

```
- (void)brandSelect:(id)sender
{
  [titleLongevity setBrand:[sender indexOfSelectedItem]];
  [titleLongevity display];
  [nbSelTitles setIntValue:[titleLongevity nbSelTitles]];
}

- (void)seriesSelect:(id)sender
{
  [titleLongevity setSeries:[sender indexOfSelectedItem]];
  [titleLongevity display];
  [nbSelTitles setIntValue:[titleLongevity nbSelTitles]];
}
```

Listing 3 bis *Extract from TitleLongevityView.m*

```
- (void)drawRect:(NSRect)rect
{
  NSString *theString;
  short i, j, startEditMonth, lastEditMonth, tlarr[600];
  // get the right array of titles
  [comicsBase sortArray:array withBrand:brand
    withSeries:series withKind:1 withState:0 withSort:0];
  nbSelTitles = [array count];
  startEditMonth = [comicsBase startEditMonth];
  lastEditMonth = [comicsBase lastEditMonth];
  // clear and then fill the local array to be graphed
  for(i=0; i<(lastEditMonth-startEditMonth); i++)
    tlarr[i] = 0;
  for(i=0; i<nbSelTitles; i++)
  {
    CTitle *thisTitle = [array objectAtIndex:i];
    NSMutableArray *theseIssues = [thisTitle issues];
    // if edited then add 1 if there is an issue for this particular edit month
    if (!editOrLong)
      for(j=0; j < [thisTitle nbIssues]; j++)
        tlarr[[[theseIssues objectAtIndex:j] editMonth]
```

```
            - startEditMonth] += 1;
  // else add 1 for all months between the first published issue to the latest
    else
      for(j = [[theseIssues objectAtIndex:0] editMonth];
        j <= [[theseIssues lastObject] editMonth]; j++)
            tlarr[j-startEditMonth] += 1;
  }
  // draw the axes
  PSsetrgbcolor(0, 0, 0);
  PSmoveto(orx, ory-3); PSlineto(orx, 700); PSstroke();
  PSmoveto(orx-3, ory); PSlineto(700, ory); PSstroke();
  // put the labels on vertical axis
  for(i=10; i<=130; i+=10)
  {
    PSmoveto(orx-3, ory+i*5);
    PSlineto(700, ory+i*5);
    PSstroke();
    theString = [NSString stringWithCString:gnums[i]];
    [theString drawAtPoint:NSMakePoint(orx-20,
    ory-12+i*5) withAttributes:dictionary];
  }
  // put the labels on horizontal axis
  for(i=0; i<(lastEditMonth-startEditMonth+13); i++)
    if (((i + startEditMonth-1) % 12) == 0)
    {
      j = (i + startEditMonth-1) / 12;
      PSmoveto(orx+i, ory);
      PSlineto(orx+i, ory-3-((j % 2)?7:0)); PSstroke();
      theString = [NSString stringWithCString:gnums[j]];
      [theString drawAtPoint:NSMakePoint(orx+i-9,
      ory-21-((j % 2)?7:0)) withAttributes:dictionary];
    }
  // draw the graph in blue
  PSsetrgbcolor(0, 0, 32767);
  for(i=0; i<(lastEditMonth-startEditMonth); i++)
  {
    PSmoveto(orx+i, ory);
    PSlineto(orx+i, ory+tlarr[i]*5);
    PSstroke();
  }
}
```

CalendarController and CalendarView

The Controller for the "Calendar" window which is a View, and the custom view inheriting from NSView which implements the tabulated display.

This is another very basic use of an NSView. The CalendarView is just some text displayed like a table, and the controller allows, again, the user to change the criteria of the display. There is no big difference from the TitleLongevityView except for one trick. At first, I filled the NSString object to be displayed with the entire text data (with carriage returns), and had only a single drawAtPoint:withAttributes: call. The result was that the lines were reversed from bottom to top (which makes sense when you remember that the origin is in the lower left corner and that the y axis is directed towards the top). Although I could have reversed the text in the NSString object, it made more sense (for easy code reading purposes) to display the text one line at a time, correctly anchored where it should be.

Listing 4 *Extract from CalendarView.m*

```
- (void)drawRect:(NSRect)rect
{
  char strtit[] =
"        OCT NOV DEC  97 FEB MAR APR MAY JUN \
       OCT NOV DEC  97 FEB MAR APR MAY JUN \
       OCT NOV DEC  97 FEB MAR APR MAY JUN \n",
    strlin[200], strstart[7];
  short i, j, k, n, lastmonth, isharr[189][9];

  // get the right array of titles
  [comicsBase sortArray:array withBrand:brand
    withSeries:series withKind:kind withState:2
    withSort:sort];
  nbSelTitles = [array count];
  lastmonth = [comicsBase lastEditMonth];
  // fill the correct month names or year value in strtit based upon
  // the model above
  for(i=1; i<=3; i++) for(j=1; j<=9; j++)
    strncpy(&strtit[6 + (i-1)*45 + (j-1)*4],
      (k = (lastmonth-1-9+j) % 12) ? gmonths[k] :
      gnums[(lastmonth-1-9+j) / 12], 3);
  // and draw it
  [[NSString stringWithCString:strtit]
    drawAtPoint:NSMakePoint(0,rect.size.height-20)
    withAttributes:dictionary];

  // clear and fill the local array of issue numbers to be displayed
  for(i=0; i<189; i++)
    for(j=0; j<9; j++) isharr[i][j] = -1;
      for(i=0; i<nbSelTitles; i++)
      {
        ... some code irrelevant to the purpose of this
            article, see the source code for details
      }
  // and display it
  for(i=0; i<63; i++)
  {
    strcpy(strlin, "");
    for(j=i; j < nbSelTitles; j += 63)
    {
      strcpy(strstart,
        [[[array objectAtIndex:j] abb] cString]);
      while (strlen(strstart) < 6)
        strcat(strstart, " ");
      strcat(strlin, strstart);
      for(k=0; k<9; k++)
        strcat(strcat(strlin,
          ((n = isharr[j][k]) < 0)?"   ":
          (char *)gnums[n]), " ");
      strcat(strlin, "   ");
    }
    [[NSString stringWithCString:strlin]
    drawAtPoint:NSMakePoint(0,rect.size.height-11*(i+3))
    withAttributes:dictionary];
  }
}
```

InputController

After getting familiar with the framework by doing some basic stuff (what you read above), I eventually reached the level where I wanted to do more ambitious and serious stuff. This Controller (for the "Input" window View) enables the user to modify the Comics Database in a lot of ways.

First, I started with an NSTableView, then implemented the willDisplayCell:forTableColumn:row: method which allowed me to control the graphical aspect (such as fonts and colors, including background colors) of a cell, before objectValueForTableColumn:row: is called.

Then I implemented the setObjectValue:forTableColumn:row: method which allows the user to type in text data in the first 2 columns (the only ones which have been set up as editable in IB).

Listing 5 *Extract from InputController.m*

```
- (int)numberOfRowsInTableView:(NSTableView *)tableView
{
  return nbRows;
}

- (void)tableView:(NSTableView *)tv
    willDisplayCell:(id)cell
    forTableColumn:(NSTableColumn *)tableColumn
    row:(int)row
{
  NSString *identifier = [tableColumn identifier];
  [cell setFont:([identifier cString][0]
    == 'M') ? fontNonProp : fontProp];
  if ([identifier isEqualToString:@"CurIsh"])
    [cell setFont:fontNonProp];
  if ([identifier isEqualToString:@"Empty"])
    if (row == [tv selectedRow])
      [cell setBackgroundColor:[NSColor redColor]];
    else
      [cell setBackgroundColor:[NSColor blackColor]];
  else if ((row == [tv editedRow])
            && ([tv columnWithIdentifier:identifier]
              == [tv editedColumn]))
    [cell setBackgroundColor:[NSColor whiteColor]];
  else
    [cell setBackgroundColor:arrCol[row % 6]];
  [cell setDrawsBackground:YES];
}

- (id)tableView:(NSTableView *)tv
    objectValueForTableColumn:
      (NSTableColumn *)tableColumn row:(int)row
{
  CIssue *thisIssue;
  CTitle *thisTitle = [array objectAtIndex:row];
  NSString *identifier

            = [tableColumn identifier], *result;
  thisIssue = [[thisTitle issues]
    objectAtIndex:[thisTitle
      findIssue:curIshArray[row]]];
    if ([identifier isEqualToString:@"Abb"])
      result = [thisTitle abb];
    else if ([identifier isEqualToString:@"Title"])
      result = [thisTitle title];
    else if ([identifier isEqualToString:@"Brand"])
      result = [thisTitle brand];
    else if ([identifier isEqualToString:@"Series"])
      result = [thisTitle series];
    else if ([identifier isEqualToString:@"State"])
      result = [thisTitle tstate];
    else if ([identifier isEqualToString:@"Kind"])
      result = [thisTitle kind];
    else if ([identifier isEqualToString:@"CurIsh"])
      result = [NSString
            stringWithCString:gnums[curIshArray[row]]];
    else if ([identifier isEqualToString:@"Grade"])
      result = [thisIssue grade];
    else if ([identifier isEqualToString:@"Type"])
      result = [thisIssue ishtype];
    else if ([identifier isEqualToString:@"Content"])
      result = [thisIssue content];
    else // column identifiers "M1" to "M6"
    {
      short k, j = -1,
        i = theEditMonth - 6
          + [identifier cString][1] - '0';
      for(k = [thisTitle nbIssues]-1;
      (k >= 0) && (j == -1); k--)
        if (  ([thisIssue = [[thisTitle issues]
              objectAtIndex:k] editMonth] == i)
              && !([thisIssue issueFlags] & mskMiss)  )
          j = k;
        if (j == -1) result = @"";
        else
          result = [NSString stringWithCString:gnums
                      [[thisIssue issueNumber]]];
  }
  return result;
}
```

At this point, I had a scrolling table with rows of different background colors, and cells with text set up in different fonts, and we could edit textually the first 2 columns.

For the other columns, it made more sense to have popup buttons to allow the user to edit the criteria by choosing rather than by typing. So I implemented a doClick: action method which I connected with IB (this name, **doClick**, is purely mine, we can call this method any name we want, the important thing is that it must be connected as a target/action method in IB for the **NSTableView**). In this method, I set up (see the next section) the popup button depending on the row and column of the clicked cell, with both the InputController and PopupInTable classes containing outlets (**puit** and **ic** respectively) pointing at each other (connected with IB as usual). Since this popup button is "kinda" floating above the **NSTableView**, extra care has to be applied to have a good user experience. For instance, it disappears when you click elsewhere: **doClick** calls **releasePub** before it sets up a new one, **_update** (called when the display criteria changed) also calls **releasePub**.

But that's not enough... If the popup button is displayed and the user scrolls the table, then the image of the popup button scrolls along, but it really still stays where it was, meaning that we get a "phantom" undesirable popup button. To have the popup button disappearing when the user scrolls is a little bit more complex. First, in **awakeFromNib**, I parsed the superview hierarchy to get the **NSScrollView**. The documentation states that there is one, but nothing is said about other superviews, and as a matter of fact, between the **NSTableView** and the **NSScrollView**, there is an **NSClipView**; since it is undocumented, I can't rely on the fact that the **NSScrollView** is 2 levels up in the hierarchy since it could change in the future, hence the little loop:

```
for ( aView = inputView;
  ![aView isKindOfClass:[NSScrollView class]];
  aView = [aView superview]
);
```

When I had the **NSScrollView**, I saved the current action method and target outlet of its verticalScroller, and I replaced them with the **userHasScrolled:** action method and self (respectively).

Then in **userHasScrolled:**, just call **releasePub** and **sendAction:** to the previously saved action and target of the verticalScroller (or else, our **NSTableView** wouldn't scroll anymore...).

***Listing 6** Extract from InputController.h*

```
@interface InputController : NSObject
{
  NSScroller *myVerticalScroller;
  id saveVerticalScrollerTarget;
  SEL saveVerticalScrollerAction;
}
```

***Listing 6 bis** Extract from InputController.m*

```
- (void)awakeFromNib
{
  NSView *aView;
  // finding the NSScrollView superview...
  for (aView = inputView;
    ![aView isKindOfClass:[NSScrollView class]];
    aView = [aView superview]);
  myVerticalScroller
    = [(NSScrollView *)aView verticalScroller];
  // saving the current target and action of the vertical scroller
  // to be able to call them later in userHasScrolled
  saveVerticalScrollerTarget
    = [myVerticalScroller target];
  saveVerticalScrollerAction
    = [myVerticalScroller action];

  // set the new target and action. Much more elegant than patching,
  // much more efficient than subclassing.
  [myVerticalScroller setTarget:self];
  [myVerticalScroller
    setAction:@selector(userHasScrolled:)];
}
```

```
- (void)userHasScrolled:(id)sender
{
  // if the user has scrolled then release the pop up button
  [puit releasePub];
  // and call back the original target and action of the vertical scroller
  // so that we actually scroll... (see below)
  [myVerticalScroller
    sendAction:saveVerticalScrollerAction
    to:saveVerticalScrollerTarget];
}
```

The number of lines of code to achieve this interception mechanism is much less than its explanation, and the target/action mechanism is the real power of the Yellow Box framework. With this mechanism, we can add new behaviors without having to subclass (and in this particular case, the subclassing would not have been trivial nor easy).

PopupInTable

This is the Controller for the "floating" popup menu button.

The setUpPopup method creates the appropriate menu items according to the row and column of the clicked cell, memorizing in a separate array the command and attribute for each menu item (easier than trying to extract the information later from the title of the selected menu item). Since I wanted some empty separation menu items in the popup, I discovered that you can't add the same title twice (only one will remain), so that's why I use addItemWithTitle:@"" and then addItemWithTitle:@" " and then addItemWithTitle:@" ", etc.

The pubSelect: action method modifies the comics database according to the choice selected by the user. The only twist is that, when the user selects the "Other..." menu item, the cell is made editable and the editColumn:row:withEvent:select: method (of NSTableView) is called; then the user has to type in the text data which is then analyzed in the setObjectValue:forTableColumn:row: method (of InputController).

To prevent endless testing of an empty issues array of the CTitle object (in nearly all of the other controllers or CComics and CTitle objects), we forbid the user from deleting the last issue of any title, suggesting deletion of the title itself instead. For the same reason, when the user creates a new title, it comes with a first issue automatically.

Since the source code for both setUpPopup and pubSelect: is way too long for me to insert in these pages, I invite you, instead, to take a look at it directly.

Disclaimer

This specific usage of a floating popup menu button as I'm using it, is strictly my own user interface. As it goes, some of my colleagues don't like it since they feel it doesn't respect all the proper user interface guidelines. Do not take this sample code as a model for your own user interface in your projects.

For speed optimization, since a reloadData method call to the NSTableView would be costly (there are a lot of rows and columns all with different fonts and background colors in our NSTableView), and since the deselection of a currently edited row forces the redisplay of that row calling objectValueForTableColumn:row: for each cell of this row, I call, where appropriate, the sequence deselectRow:/selectRow:byExtendingSelection: to always have a valid display of the edited row.

Conclusion

You'll find out, using the Yellow Box framework, that you don't have to subclass as much as you would using a C++ framework. The dynamic binding, coming from Objective-C, enables us to rely more on the concepts of target/action, delegation and notification. This dynamic binding has its advantages and its drawbacks: you can inspect objects at runtime to determine their abilities, thus use objects that you never knew about as long as they respond to the appropriate messages, reuse objects much more easily, etc. On the downside, you can't have as much strong type checking at compilation time than you would get with C++, which may lead to interesting experiences at debugging time.

As far as notifications go, the Yellow Box mechanism is very simple to use. Whenever the content of the database changes, the InputController object will send a notification this way:

```
[[NSNotificationCenter defaultCenter]
  postNotificationName: ComicsDidChange object: self
];
```

Any other Controller (Verify, Calendar, Browser, TitleLongevity), wishing to be informed has just to register for this notification, in its init method, for example:

```
[[NSNotificationCenter defaultCenter]
  addObserver: self selector: @selector(comicsChanged:)
  name: ComicsDidChange
  object: nil
];
```

then its comicsChanged: method will be called to do whatever appropriate to change the display of the window, whenever the InputController sends the notification. The only thing to remember is to unregister this notification in the Controller's dealloc method or else the Notification manager will try to send a notification to a released object.

What could be more simple?

Thanks to our technical reviewers Michelle Wyner, Deborah Grits, Andy Bachorsky, Andy Belk, Alex Dosher, Tony Frey and Randy Nelson.

develop

by Erik Sea

Preferential Treatment under Apple Location Manager

Apple Location Manager (ALM) is a suite of system software enhancements that shipped in January, 1997, but unless you're a PowerBook user, you may not have seen ALM in action. Later this year, however, you'll see a completely new version that works on any Mac OS 8 computer, be it large or small, battery-powered or tethered to the electrical grid. With increased availability, it's become even more practical to add ALM-savviness to those services, tools, and applications that need it, and over the next few pages we'll discuss some ways to do just that.

At the heart of ALM is the idea that many Mac OS users repeatedly fiddle with various control panels and change preferences. They make these changes for a number of reasons: a mobile user who travels from place to place needs to adjust to those different environments, while a home user might have different settings for telecommuting versus playing games. With ALM, the chore of going through and repetitively changing a large number of settings is reduced to a single Control Strip menu choice, once the collection of settings has been given a name. With this named collection of settings and actions, known as a "location" in ALM, lengthy lists of reconfiguration steps (some users even have these taped to their monitors or stuck to their keyboards) should become a thing of the past — provided the software they need to configure is ALM-savvy.

Should you make your software ALM-savvy? Well, if you have a product that has preferences that the user can change, directly or indirectly, the answer is yes! You might think that your software's preferences are simple — maybe not much more than a simple "on-off" or two that the user can change with a click, but, in the larger context, the user might want to turn your preferences on or off at the same time as changing something else — batch reconfiguration is a surprisingly common activity. Consider turning file sharing on or off: file sharing is only useful when there is a network around, and users might forget to turn it off at 37,000 feet, but if they've defined a location called "Air Travel" that turns it off for them (and perhaps turns off AppleTalk and dims the screen for better power conservation), they might just get a few precious minutes of additional battery life over just dimming the screen — but will they remember to do it? With ALM, they only need to remember the details when they create their locations — from then on, they need only use those locations.

You might also think that your users generally don't change their preferences, and it's true that many don't. However, you may want to consider whether they would if it were easier and could be done in concert with other changes to their environment. Some applications nest their preferences so deep in subpanels that the utility of making the change is offset by the cost of making it — but what if it were easier? I'm sure you're starting to see what I'm getting at.

There are many ways for your software to interact with ALM, including some I probably haven't thought of. I'll present some in my allotted space, but please do get in, explore, and create things that just make sense — the ideas are simple, but the uses are numerous!

Erik Sea (sea@apple.com) Erik arrived at Apple about a year ago, with the sole objective of inundating the PowerBook Division with slinkys — so far, they remain confined to his office, except on weekends when they explore the building and feed on weak or infirm paper clips. When not tweaking Location Manager code or the ALM web site, he can be found volunteering his time to rearrange sets of encyclopedias so that the volumes run from right to left on the shelf (because that's the way the pages are aligned), or writing games in COBOL on his Apple ///+.

WAYS AND MEANS

A developer once commented to me that ALM is one of the simplest, yet most complex pieces of system software to come along in a while. While you may shy from such paradoxical remarks, they correctly reflect the continuum of development options at your disposal — ALM-savviness can take many forms, ranging from tight integration with your software to standalone ALM modules written by third parties for your products.

One of the first ALM modules written outside Apple was an ALM module for Apple's OT/PPP developed, as shareware, by Prime Computing Systems and available on the web at <http://www.primecs.com>.

ALM modules are generally self-contained *preference-swappers* that cooperate to varying degrees with the software that owns the preference in question. The tighter the integration, the better the user experience, but you can certainly start out with looser coupling and improve it as revisions to your product permit. ALM modules implement some or all of the ALM module API, and are placed in the Location Manager Modules folder, inside the Extensions folder. I'll discuss module development shortly.

There are other types of ALM modules, called *action* modules, that, rather than swapping preferences for a piece of software, perform an activity that may or may not be associated with a specific piece of code, but just automate the process of *doing* some sequence of steps during a location switch: opening a file, for example. For more on action modules, and the slight differences between their API and the API of standard modules (for example, one of the "optional" calls for a standard module becomes "required" for an action module), see the ALM SDK.

To find the latest ALM SDK, go to our web site at http://devworld.apple.com/dev/alm/, or see the Mac OS SDK on CD. To take advantage of the newer features of ALM 2.0, be sure you're using the SDK's interfaces and libraries — older versions of these components may have come with your development environment, and don't have all of the new features. See "Whatcha Got?" for information on testing for the availability of features.

The ALM engine — that is, the part that provides the core services related to switching — has a public API that developers can use to enumerate the user's locations, create new locations, or switch locations. We'll cover the ALM API later.

Finally, although ALM modules can provide a great deal of flexibility to a developer who wishes to become ALM-savvy, but there are cases where a piece of software might benefit from a deeper knowledge of what the engine is doing. There is the API, as mentioned earlier, but you should know that it can be used in clever ways that might be appropriate for some software that would benefit from even tighter integration with ALM. A hypothetical example of this kind of location-awareness, going beyond mere ALM-savviness, is discussed in the latter part of this article.

WHATCHA GOT?

As with any system software from Apple, there are means for testing what features are present in ALM, since the features have changed from version 1.0 to 2.0.

gestaltALMVers

This selector returns the version of the ALM engine that is installed on the machine.

gestaltALMAttr

This selector returns a bitfield representing engine functionality with the following bit meanings:

- gestaltALMPresent is on if ALM is installed; it's possible, on some hardware, for the selector to be defined but for this bit to be off, so check!
- gestaltALMHasSFLocations is on if the Get, Put, and Merge location calls are implemented.
- gestaltALMHasCFMSupport is on if the engine recognizes CFM-based modules, otherwise only Component Manager modules are directly supported.
- gestaltALMHasRescanNotifiers is on if notifications for changes in the names or number of locations will be sent via callbacks and Apple events; otherwise, only notifiers for switches are generated.

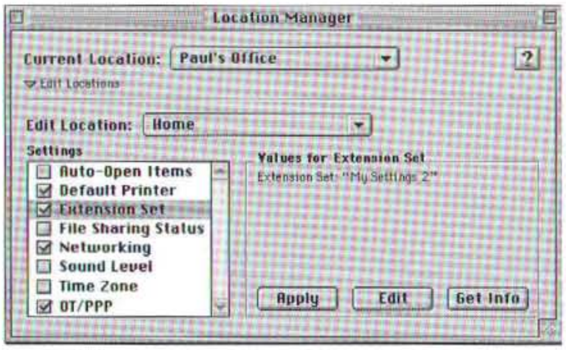
With so many options to choose from, it might take a while to figure out how your software and ALM might collaborate. So, as you dive into the rest of this article, keep in mind that there is often more than one path to ALM-nirvana, er, savviness, but the sooner you start, the sooner you'll arrive. Setting aside philosophical opinion, let's look at the path most traveled first.

MAGNIFICENT MODULES

For most situations, the ALM module provides the easiest and best-encapsulated way to get your software plugged into the ALM engine. The calls that an ALM module supports are very general: a module (or any software it interoperates with) doesn't need to know anything about locations or how the ALM control panel works, just how to interact with the software it represents.

JUST WHAT IS A MODULE ANYWAY?

It is difficult for me to describe what an ALM module is or does without showing how the user interacts with them. To define a location, the user runs the ALM control panel, and picks "New Location" from the File menu. All location editing is done from the main window, seen in **Figure 1**. In the top part of the window, we see the current location (that is, the location a user switched to in the past or the location the user is "in" right now) specified by a pop-up. The user can switch from this pop-up or switch from a similar menu in the control strip.


Current Location: Paul's Office

Edit Location: Home

Settings
- ☐ Auto-Open Items
- ☑ Default Printer
- ☑ Extension Set
- ☐ File Sharing Status
- ☑ Networking
- ☐ Sound Level
- ☐ Time Zone
- ☑ OT/PPP

Values for Extension Set

Extension Set: "My Settings 2"

Figure 1. *Apple Location Manager main window, in expanded mode.*

The disclosure triangle toggles display of the bottom half, where a second pop-up shows the name of the location to be edited. As you can see, it's possible to edit locations that are not the current location, which creates some special issues you might have to deal with. More on that later.

At the left side of the window, we see a list of modules that ALM knows about — they are located in the Location Manager Modules folder. Each of these modules has a value associated with the edit location, and a description of that value is displayed at the right when the associated module is selected at the left.

Modules are ordinarily listed in alphabetical order, and the list order is the order that the engine applies changes when switching. The user can, using menu options, move modules around in the list to achieve different results (it's often handy, for example, to have modules that might require a restart, such as Extension Set, first, so that other modules, such as Auto-Open Items, could decide to defer execution until after the restart).

Modules could also be in the list several times if the user has chosen to duplicate them (another menu option). For example, a user might have several instances of Auto-Open Items sprinkled throughout a location definition, doing everything from mounting servers to executing AppleScript scripts.

The value for a module within a location is saved whether the module is on or off — this allows the user to temporarily alter the location by excluding a module's setting without losing the settings established for it.

Only a few calls are necessary to support everything seen in the control panel.

If you're concerned about supporting ALM 1.0, rest assured that, despite the vast difference in appearance of the control panels, ALM 2.0 calls modules with the same expectations as its predecessor, but if you want to work under both versions, you'll have to take care not to use any 2.0 features. The bottom half of the new control panel corresponds to the separate Edit Location window in the old control panel, and the new "on-off" checkbox corresponds to "adding" a module to a location in 1.0. There's some information displayed in the old interface that isn't anywhere in the new one for simplicity and performance reasons.

What a Module Needs to Know to do

Essentially, an ALM module only needs to know how to interpret the settings of the software it supports, how to change those settings, and how to get those changes recognized. We'll discuss nuances in a moment, but here are the calls a module absolutely must handle (for both ALM 1.0 or 2.0).

```
pascal OSErr GetCurrent (Handle setting);
```

When a module receives this request, it resizes the handle to whatever size is necessary to store the current setting. It is generally a good idea to "version" this information, in case you decide to change it later. In addition to the version you should also store in the handle enough information to restore the setting when ALM calls your module with the handle at a later time. This handle is stored on disk in a location file, so be sure you've "flattened" it out (that is, there are no pointers to things in memory that might not be in the same place when your module is next called).

```
pascal OSErr SetCurrent (Handle setting, ALMRebootFlags* flags);
```

Modules receive this call during a switch. The setting will be one which your module has previously returned from a GetCurrent call, though, obviously, it may have been stored some time ago. Be aware that, because ALM allows a user to switch location very early in the boot process, you need to check for things that may not be around yet (such as the Process Manager). If the environment is too hostile for your tastes, your module should return kALMDeferSwitchErr in response to this call, and ALM will try again later in the boot process.

The flags parameter is used to communicate to the engine what the impact of the setting change was, so that the user can be offered the opportunity to restart, if necessary. Possible responses include kALMNoChange if the settings did not actually need to be changed, kALMAvailableNow if they can be used immediately, or kALMExtensions if a restart is required (in which case the user will be given the option to reboot their machine) — see the SDK for additional values and their meanings. Note that the input value of the flags parameter is the current escalation of the switch process, as set by previous modules. If the reboot level is already too high, you can set the flags to whatever value is appropriate for what you actually did, and return kALMRebootFlagsLevelErr as the function result.

The input value of flags in ALM 1.0 is always kALMNoChange, and there is no way to determine what the reboot escalation level is when you are called. You shouldn't return kALMRebootFlagsLevelErr if the input escalation level is kALMNoChange unless you know you're under 2.0.

You should avoid forcing the user to reboot when they switch your settings — your customers will thank you for saving them time and trouble. Possible strategies include defining an Apple event or other signal that your software will recognize as "read your preferences now", then send that signal from your module's SetCurrent function.

```
pascal OSErr CompareSetting (Handle setting1, Handle setting2,
                    Boolean* equal);
```

This call is made to determine whether two settings represent the same values from the point of view of the module. This is used at different times and in different ways, and has a special interaction with the optional EditSetting call, discussed later.

You'll find that this call is made less often under ALM 2.0 than it was under the older version, because of UI changes, but ALM 2.0 still requires it.

```
pascal OSErr DescribeSetting (Handle setting, CharsHandle text);
```

Generate a textual description for one of your settings, resizing the CharsHandle as appropriate to fit the text. Don't go overboard in your description — the window containing the description is resizeable, but try to fit your description in the minimum window size for best results. The script system used will be that returned by the GetScriptInfo call, described shortly.

```
pascal OSErr DescribeError (OSErr lastErr, Str255 errStr);
```

You are encouraged to return your own (positive) error response values from a module API call, and you'll get first crack at telling the user what they mean. If you don't wish to describe an error, return paramErr in response to this function, and the control panel will present it to the user as a number (boo, hiss).

```
pascal OSErr GetScriptInfo (ALMScriptManagerInfoPtr info);
```

This call, though technically optional, is pretty easy to implement, and is recommended if you expect your module to be used worldwide. The format of info is as follows:

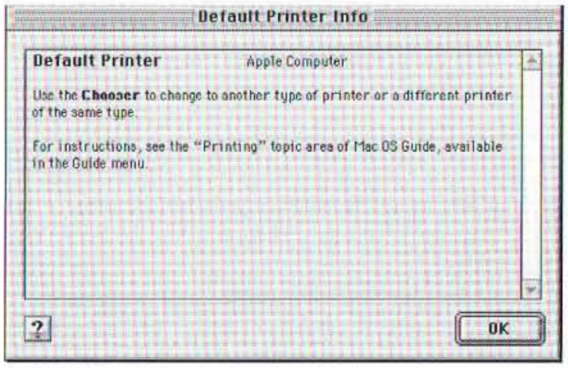
```
typedef struct {
    SInt16   version;
    SInt16   scriptCode;
    SInt16   regionCode;
    SInt16   langCode;
    SInt16   fontNum;
    SInt16   fontSize;
} ALMScriptManagerInfo;
```

You should set the version to kALMScriptInfoVersion, and fill in the font and script manager information in accordance with how you wish your text to be displayed. If you don't do this, you might not play well in other languages, because the system will fill out the script information on your behalf, and your module might end up getting displayed in the wrong script — certain characters do not translate well, to say the least. It is actually possible for modules to be of different script systems yet be displayed correctly. If you plan to localize your text, you'll probably want to put script information in a resource that gets localized at the same time. See the sample code in the SDK for a good strategy to do exactly that.

```
pascal OSErr GetInfo (CharsHandle* text, STHandle* style,
                ModalFilterUPP filter);
```

The GetInfo call is made to tell the user what your module is all about, such as what other software it interacts with. There are essentially two flavors to this call: in the first, you simply return styled text using the first two parameters, and it gets displayed, as shown in **Figure 2**. The text can be any length, but don't write a book — say what your module does, what software it works with, and where to go for more information.

In the second flavor of GetInfo, the module itself takes care of displaying whatever dialog it chooses, or perhaps opens an AppleGuide window, and returns NULL in the first two parameters to signal that it did the job on its own.



Default Printer Apple Computer

Use the **Chooser** to change to another type of printer or a different printer of the same type.

For instructions, see the "Printing" topic area of Mac OS Guide, available in the Guide menu.

Figure 2. *Control Panel Displays Results of GetInfo Call.*

MODULES OR APPLESCRIPT?

At first glance, it may seem that ALM modules and AppleScript are two technologies vying for the same role, for if every preference-generating piece of software were scriptable, an intelligent user could just have a collection of scripts to effect a location switch.

If you look deeper, though, you'll see that these technologies are complementary: a totally scriptable control panel, for example, even if it were recordable, would benefit from the centralized, simplified process of location definition and aggregation with other settings. At the same time, it is much easier to support ALM in a control panel that has already been made scriptable because the requisite code factoring makes ALM support a breeze.

Still, you do not need to be scriptable to support ALM, and, though I would prefer that you did make your software scriptable, you don't need to do it for ALM.

The filter parameter to the GetInfo call is a legacy parameter, used only in ALM 1.0. ALM 2.0 does not require you to call filter explicitly, but you may do so.

Those are the basics; the SDK provides a good sample implementation you can start from, in the form of the "Generic" module, which is particularly handy for prototyping.

In fact the Generic module could theoretically be used with only a few changes in resources, and, for some limited-use applications, or for prototyping, that might be enough, but it's probably not up to the quality required for commercial or even shareware release.

Though these calls are enough for ALM to work correctly and predictably, there has yet to be an API devised whereby you can become "savvy" by supporting just the lowest level, so let's go up a step.

DUALING SETTINGS

Over the last little while, people have begun to recognize the value of storing named configurations, or sets of preferences, for easier retrieval by users. This is a good thing, and is a move in the direction of ALM. An ALM module that deals with a piece of software that already supports named configurations probably just needs to track the name, and that makes life much easier, because you basically are dealing with just a reference to other data, rather than the other data directly.

There are, however, complications, such as the need to define dual formats for your settings: normal (in which case a reference to other data on the current machine is adequate) and import-export (in which case you need to recreate the entire setting for use on another machine).

Another question is what happens if the user changes the named configuration — will your module know how to track the change and update its reference?

And how will your module help Bob import "My Reactor Settings" from Anne's computer, without wiping out his existing (and potentially different) "My Reactor Settings"? Maybe he really wants to replace them, but maybe he wants to rename the import as "Anne's Reactor Settings".

You'll have to solve the reference-tracking problem for yourself, but ALM does provide a call to handle import name collisions, called ALMConfirmName, discussed in the API section.

Of course, if your setting is very simple, such as on or off, both formats of your setting can be the same, and your implementation of ImportExport becomes very simple — just return noErr.
The remaining bits are not currently used.

WHAT A MODULE SHOULD KNOW TO DO

With some additional work, you can provide the user with additional ALM-supported functionality and achieve ALM-savviness. The capabilities you'll need to tackle are editing of inactive settings, and import-export. If you don't implement these functions to at least some extent, not only will you fail to enter the blissful state of ALM-savviness, but your customers may become confused and you'll have a recurring dream in which you read Inside Macintosh: AOCE Application Interfaces word for word, from cover to cover, until your conscience makes you support these added functions.

This setting must change As mentioned earlier, it's possible for a user to edit the settings from one location while being in another. Through use of the "Apply" button (refer to Figure 1), the user can change a setting in a location by capturing the current value of the setting on the computer. This is done through the GetSetting call described previously. In user studies, we found that it's highly desirable for the user to be able to change settings in an inactive location without changing the live system settings (we do provide a way to reassert the current location's values). Unto this desire, the EditSetting call was born.

```
pascal OSErr EditSetting (Handle setting);
```

The EditSetting call is optional, but very strongly recommended. The input setting parameter is one that will have come from an earlier call to GetCurrent. Do whatever you need to do to edit the setting, and return either the same one (if the user cancels the edit) or a different one (if the user accepts the edit). When I say different, I mean different enough that your CompareSetting call will say, "Yes, they're different" — if there was no real change, your module might just flip a bit in the setting handle that it would ignore everywhere but on a CompareSetting call. The reason for requiring these settings to seem different even if they aren't is a subtle bit of UI trickery in the ALM control panel: the control panel will automatically turn a setting on if, as a result of an EditSetting call, it can be ascertained that the settings were accepted by the user. The control panel makes the determination of acceptance based on the input and output setting handles being different.

Implementing the EditSetting call can be done in a number of ways. Ideally, an EditSetting call would bring up the same interface you use in your software to edit your live preferences, but without affecting those live preferences — a sort of "preference UI factoring", if you will. Alternatively (but less desirably), you can take some sort of middle road, ranging from explaining to the user where they should go to edit their live system settings (and reminding them to put those settings back the way they were), to providing a little "platform" from which to launch your software, perhaps under AppleScript control.

The EditSetting call was supported under ALM 1.0, and, in fact, it's required for action modules. There is, alas, no support for EditSetting of standard modules in the old control panel, so be aware of that limitation if you plan to support both versions of ALM.

If you don't implement EditSetting, the ALM control panel has no way of knowing where your setting comes from, so it displays the message shown in **Figure 3** when the user tries to edit your setting.

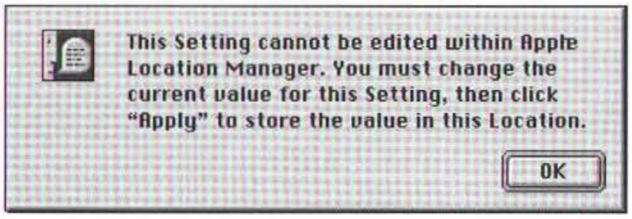

This Setting cannot be edited within Apple Location Manager. You must change the current value for this Setting, then click "Apply" to store the value in this Location.

Figure 3. *Developer Raspberry: Module does not support EditSetting.*

I can't encourage you strongly enough to implement EditSetting to some extent in your initial release, and then refine it over time (if you don't go all the way up front). I've seen hours of videotaped user experiences, and the EditSetting capability more closely reflects how users expect the machine to behave, so don't let them see this brick wall of an alert when they use your module.

Duties Paid: Importing and Exporting

It would be really nifty for a user if, once they've honed a location to near perfection, they could share it with others. It would also be nice if, walking into a new environment, a user could receive a group of settings that other people in that environment use. That, in a nutshell, is what the ImportExport capability is all about.

```
pascal OSErr ImportExport (Boolean import, Handle setting,
                           SInt16 resRefNum);
```

The settings dealt with by most of the API can be small — they may just be references to other data structures (I would encourage this design decision, though it introduces some new issues as discussed in "Dualing Settings"), and that will work fine on one user's computer. If you were to export just the name, however, the importing machine might not be able to interpret it correctly.

So, when you export, you might want to augment the setting handle with additional information that would allow you to replicate the setting on another machine (short of actually installing software on that machine!), and, on import, you can decode that information and shrink the handle back down to size. If that technique isn't enough, or you'd rather keep your setting the same, you can add resources to the exported location file, and read them in on import (after you read your resources in, you should delete them, since you're actually working on a copy of the import file, in the Locations folder, rather than the actual import file). You should use a unique resource type, such as your creator, for the resources you use.

If you don't implement ImportExport, be sure to document this in your GetInfo window information.

Now that you know how modules work and how ALM calls them, let's go to the other side of the fence and see how to call ALM.

START YOUR ENGINES!

The ALM API contains a series of calls for the developer to find out about a user's locations, a call to initiate switches, calls to create or let the user merge a setting into a location, plus events and notifications to determine when a switch has occurred or a location has been created. As mentioned previously, there is also a call to handle import collisions in a uniform way.

THE API

The ALM API is accessible in C, Pascal, or Assembly, from PowerPC or 68K, in classic or CFM-68K runtime models.

Since CFM-68K calling isn't available under ALM 1.0 I recommend weak-linking with the CFM-68K stub library and then testing against API symbols at launch.

What place is this?

If you want, you can take a gander at what the user has set up. Perhaps you want to build your own pop-up menu to provide the user the ability to switch locations from within your application, rather than the control strip or control panel. Here are the calls, and a very simple program to use them can be found in Listing 1.

```
pascal OSErr ALMGetCurrentLocation (SInt16* index,
                                    ALMToken* token,
                                    ALMLocationName name);
pascal OSErr ALMCountLocations (SInt16* locationCount);
pascal OSErr ALMGetIndLocation (SInt16 index,
                                ALMToken* token,
                                ALMLocationName name);
```

The indices in these calls are zero-based, so the maximum will be one less than the locationCount. The special index value kALMNoLocationIndex denotes the situation where the user has

Listing 1 *Printing the User's Defined Locations*

```
ALMToken            curLocToken, locToken;
ALMLocationName     curLocName, locName;
SInt16              numLocs, locIndex;

printf ("User Locations on this Machine:\n");
err = ALMCountLocations (&numLocs);
printf ("There are %d locations in total\n", numLocs);
err = ALMGetCurrentLocation (NULL, &curLocToken,
            curLocName);
printf ("The active location is %s\n",
            p2cstr (curLocName));
for (locIndex = 0; locIndex < numLocs; locIndex += 1) {
  err = ALMGetIndLocation (locIndex, &locToken,
            locName);
  if (locToken != curLocToken) // special treatment!
    printf ("Location %d is %s\n", locIndex,
            p2cstr (locName));
  else
    printf ("Location %d is the current location\n",
            locIndex);
} // for
```

chosen the special location "None (Off)". ALMTokens represent references to locations for use in other calls (and the special value kALMNoLocationToken corresponds to kALMNoLocationIndex). If you only want certain parameters to be returned, it's legal to pass NULL for any pointer.

Snapshots — as Pretty as a Picture

In ALM 2.0, there are functions that you can use to interact with the user much like the control panel does. You should gestalt these functions (see "Whatcha Got?") before using them. The user experience is very much like the Standard File package, only simpler, but the calls themselves take care of a lot of additional work for you.

Sometimes, you'll want the user to simply choose a location from a list without need for iterating across all installed locations and building your own UI. To ask the user for a location, use the ALMGetLocation call, which presents a dialog as shown in **Figure 4**.

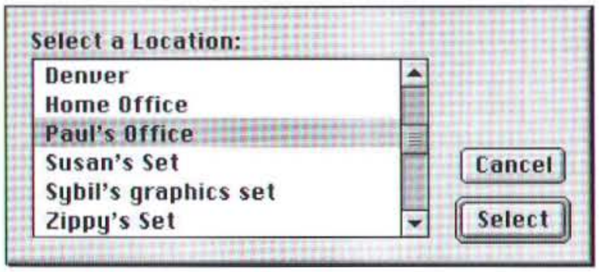


Figure 4. *Asking the user for a location with ALMGetLocation.*

```
pascal OSErr ALMGetLocation (ConstStr255Param prompt,
                    Str31 locationName,
                    ModalFilterYDUPP filter,
                    void* yourDataPtr);
```

The prompt is displayed above the list of locations, as in Standard File. The locationName is both an input and an output parameter — if the name matches a location, that location will be selected in the list. The filter and data pointer are for your own use, again as in Standard File. If the user clicks cancel, the function result is userCanceledErr; otherwise, it should be noErr.

The next two calls, ALMPutLocation and ALMMergeLocation are intended to allow an application, such as a setup program, to capture the current system settings for a number of modules as a location, or merge them into an existing location. Detailed information on module signatures, or types, is beyond the scope of this article (see the SDK), but the use of some special values for these parameters is shown in Listing 2.

```
pascal OSErr ALMPutLocation (ConstStr255Param prompt,
               Str31 locationName,
               SInt16 numTypes,
               ConstALMModuleTypeListPtr typeList
               ModalFilterYDUPP filter, void* yourDataPtr);
pascal OSErr ALMMergeLocation (ConstStr255Param prompt,
               Str31 locationName,
               SInt16 numTypes,
               ConstALMModuleTypeListPtr typeList,
               ModalFilterYPUPP filter, void* yourDataPtr)
```

Listing 2 *Put Up or Merge Up*

```
// create a new, empty location
err = ALMPutLocation ("\pName your place",
           "\pEmpty Location", kALMAddAllOff,
           NULL, NULL, NULL);

// add to a location as a snapshot, using all currently-installed
// non-action modules, replacing current definitions
err = ALMMergeLocation ("\pRedefine your place",
           "\pSnapshot Location",
           kALMAddAllOnSimple, NULL, NULL, NULL);
```

Switching

There would be very little use to ALM without the switch engine. This is the call that the control strip and control panel use to effect a change in location:

```
pascal OSErr ALMSwitchToLocation (ALMToken newLocation,
                           ALMSwitchActionFlags switchFlags);
```

Most of the time, you can pass kALMDefaultSwitchFlags for the second parameter. For a quiet switch (that is, one in which the switching dialog does not come up), pass the mask kALMDontShowStatusWindow. If you're making this call from an unusual place such as inside a dialog filter or some other weird place, use the mask kALMSignalViaAE so that the switch will be deferred until the ALM control panel (or the Finder, if the control panel isn't open) is in a better synchronized state.

If you wish to switch locations from a background application you should use either kALMDontShowStatusWindow or kALMSignalViaAE or both. If you do not, ALM 2.0 will return an error. ALM 1.0 is less forgiving, and will crash if the host application hasn't initiallized the window manager. Moral: don't use default flags except in direct response to a user action in your foreground application!

Noticing Switches and Other Goings On

One of the more interesting things about ALM is that it actually tells all running processes when the user has changed locations. Under ALM 2.0, it'll also advise when the user has created, renamed, or deleted a location, so that applications that want to know can rebuild any list of locations they might have internally, or reorganize any data they might have been shadowing on a location-by-location basis by rescanning the user's installed locations using a technique like that of Listing 1.

You should check gestalt for the availability of the rescan notifiers if you're going to rely on them, as described in "Whatcha Got?". If they are not available, you can employ alternate strategies, such as periodically iterating through the installed locations. In any case, it's good practice to recheck the installed locations at start-up if you save your own list somewhere.

***Listing 3** Observing Switches and Handling Rescan Notifiers*

```
pascal OSErr MySystemConfigNoticeHandler
                (const AppleEvent* inEvent,
                AppleEvent* reply, SInt32 refCon)
{

  OSErr         err = errAEDescNotFound;
  ALMToken      locToken;
  DescType      actualType;
  Size          actualSize;

  // Handle any other cases you like...
  [...]
  // Did we get a switch notification?
  if (err != noErr) {
    err = AEGetKeyPtr (inEvent,
            kAELocationChangedNoticeKey,
            typeInteger, &actualType,
            &locToken, sizeof (locToken),
            &actualSize);
    if (err == noErr) MyDoReactToSwitch (locToken);
  } // if
  // Did we get a rescan notification?
  if (err != noErr) {
    err = AEGetKeyPtr (inEvent,
            kAELocationRescanNoticeKey,
            typeInteger, &actualType,
            &locToken, sizeof (locToken),
            &actualSize);
    if (err == noErr) MyDoReactToRescanNotice
(locToken);
  } // if

  return err;

} // SystemConfigNoticeHandler
```

The class of the Apple event is kAECoreSuite, and the event ID is kAESystemConfigNotice, just as with the display manager's notifications that the screen size or resolution has changed. If your event handler receives such an event, and there is a parameter keyed under kAELocationChangedNoticeKey, then a switch has occurred, and that parameter's value is the ALMToken of the location to which the user switched. If there is a parameter under kAELocationRescanNoticeKey, then the user has made changes to the installed locations, and you can't rely on any of your previously stored data about locations being valid. The value of the parameter is the ALMToken of the current location. For a skeletal handler, see Listing 3.

Event Me Not

Of course, if you're a code resource, you won't be getting many Apple events, so you can provide a function such as this as an alternative:

```
pascal void MyALMNotificationRoutine (AppleEvent* theEvent);
```

Your implementation of this routine ought to be similar to the Apple event handler in Listing 3. Create a UPP by passing your routine to NewALMNotificationProc, and then register and deregister your routine using these calls:

```
pascal OSErr ALMRegisterNotifyProc
                (ALMNotificationUPP notifyProc,
                const ProcessSerialNumber* whichPSN);
pascal OSErr ALMRemoveNotifyProc
                (ALMNotificationUPP notifyProc,
                const ProcessSerialNumber* whichPSN);
```

The ProcessSerialNumber parameter is used by ALM to ensure that notifications aren't sent to code whose process no longer exists (but for which the notifier was not removed). You should pass the serial number you're associated with.

Import Collisions

Particularly if you implement named configurations, as discussed in "Dualing Settings", you could have situations, when importing, where a setting you're importing conflicts with a setting already on the machine. To address this problem, ALM provides a routine that displays two dialogs in succession. The first asks the user to choose either to rename or to replace the named configuration, and the second, shown in case of a rename, presents a box in which to rename the configuration. The call is as follows:

```
pascal OSErr ALMConfirmName (ConstStr255Param msg,
                Str255 configName,
                ALMConfirmChoice* choice,
                ModalFilterUPP filter);
```

If the user cancels the operation, the function returns userCanceledErr; otherwise, the value of choice is kALMConfirmRename or kALMConfirmReplace, depending on the user's choice. You may provide a filter function, if you wish access to events in the dialog. The refcons of the dialog boxes disclose which is which, and the individual element numbers are provided as constants in the ALM 2.0 interfaces, should you wish to do things like constrain the allowed characters in your configuration name (rather than validating the name after the call returns). For details on the refcons and item numbers, see the SDK.

***Listing 4** Driving a Switch from AppleScript*

```
set someLocationName to "Home Office"
tell application "Finder"
  set cpFolder to (control panels folder as string)
  open file (cpFolder & "Location Manager")
  tell application "Location Manager"
    set current location to location someLocationName
  end tell
end tell
```

Due to a bug in ALM 1.0 the filter function was not called in both dialogs, but just the rename dialog. If your filter function sees a refcon of zero, you can assume you are in the 1.0 rename dialog box.

AppleScript Support

Alternatively, you can access much of the API by sending AppleScript commands to the control panel. A simple switch through AppleScript is shown in Listing 4. You can, of course, also list locations from AppleScript using the Standard Suite.

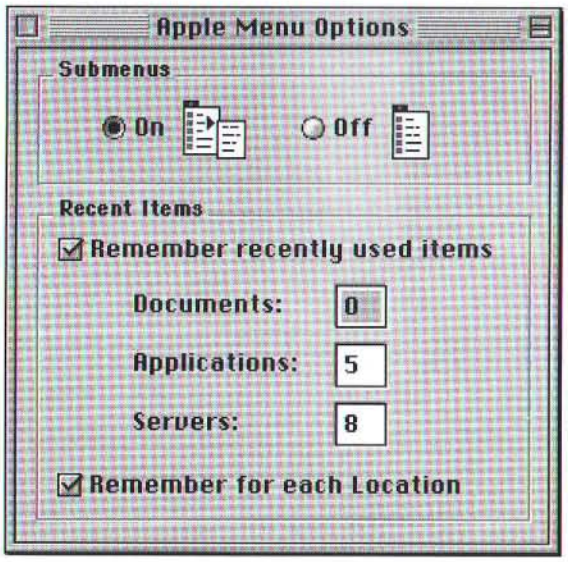
Though this API is small, these same bricks will build a house or a castle, depending on how you choose to use them. Although the water may look deep, location-awareness could represent the ultimate offering to the user, giving them much more than ALM modules, alone, are capable of.

Where Am I?

Thus far, we've talked about ALM-savviness and driving the engine, but now let's switch gears a little and talk about the greater generalities of location-awareness.

How a piece of software becomes location-aware is as unique as the piece of software itself — it requires planning and thought beyond how to simply automate the chore of switching preferences. Instead, a location-aware piece of software has new functionality that its location-oblivious sibling does not.

As an example, let's take the Apple Menu Items control panel, which has been an indispensable part of Mac OS from System 7.5 to the present. What form could location-awareness take in this control panel (see **Figure 5**)? Suppose that, in each location (whether a "location" is a physical location or merely a state in which they perform different tasks), the user accesses different documents, applications, and servers that are specific to that location, but, in moving from one location to the next, they constantly wipe out recent items from one place before they get back to it. Typically, users faced with this problem combat it by increasing the recent item allocation numbers, but while searching through a big list works, it's far from ideal.



☑ Remember recently used items

Documents: 0

Applications: 5

Servers: 8

☑ Remember for each Location

Figure 5. *Mock-Up: Apple Menu Items with Location-Awareness.*

We could just write a module for this that, through clever folder manipulation, swapped the recent items around, but what about the occasion when the user actually wants to access a recent item from another location without switching to that location first? Perhaps the submenus could present recent items in different ways — the ordinary way most of the time — or sorted by location if the user holds down the option key when choosing the Apple menu.

I am not suggesting that the hypothetical example I present above is the best possible answer, but I am hoping that software developers will consider using location-awareness as a new way to solve problems that their users might be facing.

Give location-awareness some thought — it might be your best bet.

There's no Location like Home

Now that you've been through savviness and awareness, modules and APIs, I hope you can see the opportunities ALM presents for you to make life easier for your users, both mobile and landlubber.

So take a few moments to install and play with ALM, and then play with the sample code. I think you'll be surprised at how much mileage you can get out of a very small amount of labor!

Thanks to my reviewers Mark Cookson, Dave Ferguson, Scott Johnson, Susan Michalak, Kent Miller, and Eric Slosser. Special thanks to everyone on the ALM team, present and past, for your contributions to shaping and sculpting a truly unique and innovative piece of software. MT

RELATED READING

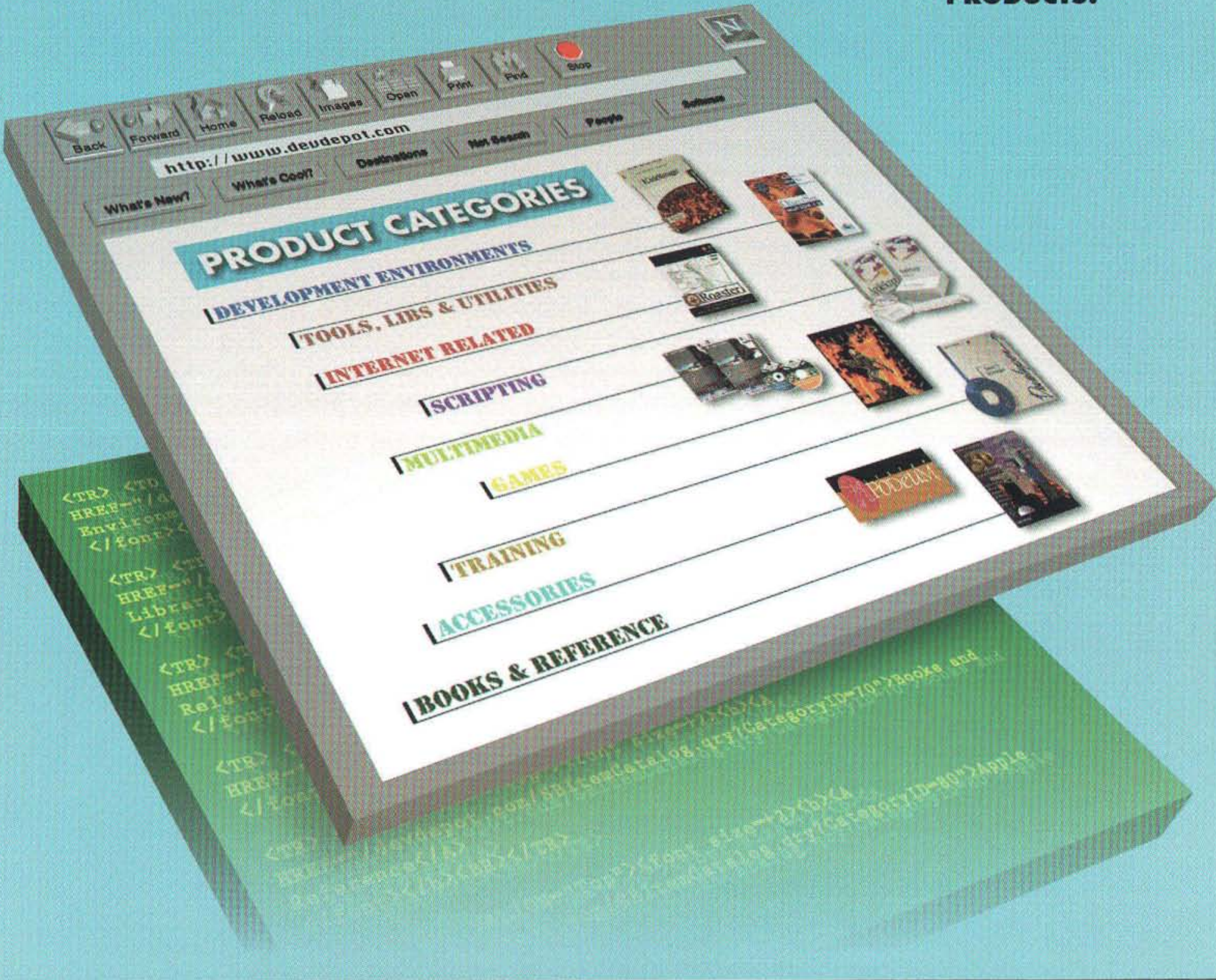
- The ALM web page, located at http://devworld.apple.com/dev/alm/.
- ALM SDK, available through the web site or on Mac OS SDK.

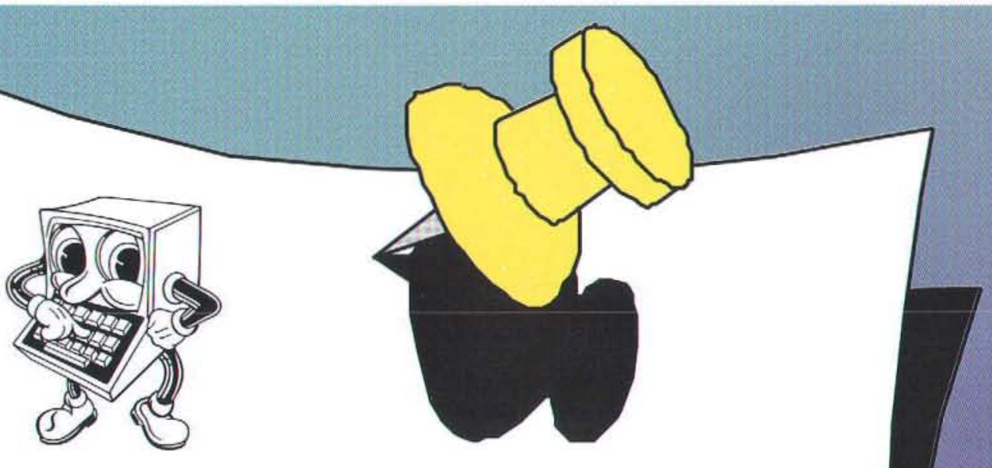
Developer DEPOT™

Developer Depot PO Box 5200 Westlake Village CA 91359-5200
800/MACDEV-1 • Outside the U.S. & Canada: 805/494-9797 • Fax: 805/494-9798
E-mail: orders@devdepot.com • Web Site: http://www.devdepot.com

- ✓ World renowned customer service
- ✓ Order by phone, E-mail, fax or through our continually updated Web site
- ✓ Hundreds of developer products
- ✓ Satisfaction and lowest price guaranteed

INCLUDES EVEN MORE APPLE BRANDED DEVELOPMENT PRODUCTS!





Developer To Do List:

- [✓] Prioritize project list & generate schedule
- ☐ Pick up new MacTech CD-ROM v. 1-12
- ☐ Check out all the Dev Depot specials
- ☐ See Apple Branded Tools at Dev Depot
- ☐ Tell the gang about Dev Depot deals
- ☐ Stock up on the latest books & utilities
- ☐ Enter Dev Depot info into contact file

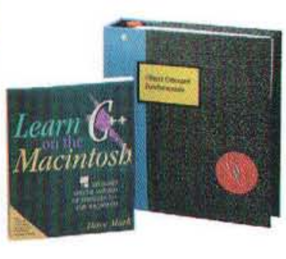
Developer Depot
PO Box 5200
Westlake Village, CA 91359-5200

Voice: 800/MACDEV-1
Outside US/Canada: 805/494-9797
Fax: 805/494-9798
E-mail: orders@devdepot.com
http://www.devdepot.com

Developer DEPOT™

ANNOUNCEMENT

Developer DEPOT™ now carries Apple brand products



QuickTime VR

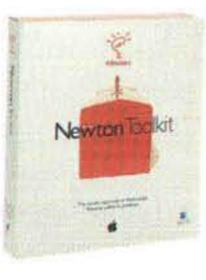


TABLE OF CONTENTS

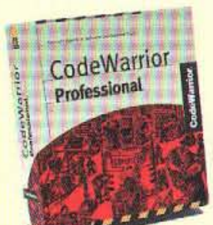
Lowest price guaranteed. Feel Good about your purchase! 30-day return policy.

One & 2-Day SHIPPING (call for details)

Order Toll-free 800-MACDEV-1 (800-622-3381)

Developer Depot 30 day Money Back, Price and Satisfaction Guarantee

Developer Depot products are sold with a 30 Day money back guarantee on user satisfaction, lower prices and against defects. If, for any reason, you are not satisfied or find the same product at a lower price within 30 days, please call Customer Service at 800-MACDEV-1 and request a Return Merchandise Authorization (RMA) number to get a full refund or the difference in price (where applicable). You must return undamaged product at your expense, including all its original packaging, documentation and the blank warranty card if applicable. Developer Depot will replace defective product upon receipt of the defective merchandise. Please remember to back up your data before installation of any new hardware, software, or peripherals; we cannot be responsible for any lost data. Policies, item availability, and prices are subject to change without notice. The price in effect when we receive your order will be the price that you are charged. We are not responsible for any typographical errors in this or any other catalog, nor for any misstatements from any vendor. Purchase orders are not accepted without prior approval. Call for more information.



NEW PRODUCT!

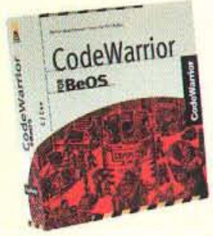
CodeWarrior Professional Release 1

by Metrowerks

More platforms, more languages, more options: CodeWarrior Professional is designed to give you the tools you need for serious, industrial-strength programming. CodeWarrior Professional is the only Integrated Development Environment (IDE) in which you can edit, compile and debug C, C++, Java and Pascal programs for multiple target processors and operating systems. CodeWarrior's compilers produce fast, highly optimized code for Windows 95/NT running on x86, or Mac OS running on 68K or PowerPC processors. CodeWarrior Professional features the new CodeWarrior IDE Version 2. The newly revamped Project Manager supports multiple projects open simultaneously, multiple targets per project, and threaded execution, plus it's significantly faster. CodeWarrior Professional also includes online books, documentation, and reference materials, as well as tutorials and sample code. We support your development efforts with one free update and free world-class technical support for a year with registration.

- Includes Windows 95/NT and Mac OS versions of the CodeWarrior IDE
- Supports C, C++, Java and Pascal
- Develop for Windows 95/NT on x86 and Mac OS on 68K/PowerPC
- New Project Manager supports multiple open projects, subprojects, multiple targets per project, and threaded execution
- Integration of tools means you spend less time navigating and more time coding

Upgrade for competitive product owners
(SCWPRUP) Our Price **$449**
Full Version
(SCWPRO) Our Price **$599**



Be OS™

CodeWarrior for BeOS 3

by Metrowerks

- Start programming for the new, innovative Be Operating System (BeOS) with complete set of Codewarrior tools
- BeOS-native Integrated Development Environment (IDE) with all the familiar CodeWarrior features at your fingertips
- A BeOS PowerPC compiler and linker, an editor w/syntax color and styling, and a source-level debugger
- BeOS header and libraries, complete documentation, useful C++ classes, and sample code
- The Be Operating System Advanced Access Preview Release allows you to run and program for the BeOS on a 603 or 604 PCI based PowerMac

(SCWFB) Our Price **$299**

metrowerks

NEW PRODUCT!

CodeWarrior for PalmPilot

by Metrowerks

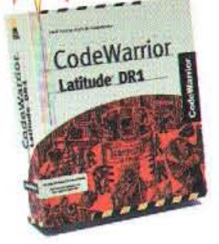
CodeWarrior for PalmPilot is the first complete set of tools which enables you to develop for the U.S. Robotics PalmPilot connected organizer, from the comfort of your PC or Macintosh computer. All the tools you need are included: the award-winning CodeWarrior IDE, compiler, linker, source-level debugger, GUI builder, and other tools, plus online documentation and reference materials. Includes one year of product updates and free technical support.

(SCWPALM) Our Price **$369**

NEW PRODUCT!

CodeWarrior Latitude

by Metrowerks



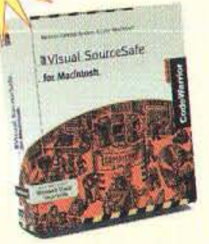
Don't throw away the investment you have made in your Mac OS applications! With the DR/1 release of CodeWarrior Latitude, you can prepare your Macintosh applications to run native under Silicon Graphics IRIX or Sun Solaris (and soon, Rhapsody). Begin work now to prepare your Mac OS application to run native under Rhapsody. By compiling and linking your CodeWarrior project with the Latitude libraries, you can identify which portions of your code will port smoothly and which Mac Toolbox traps are not implemented. Additionally, you can expand the market for your Mac OS application by completely porting it to Silicon Graphics IRIX or Sun Solaris at a fraction of the cost of other porting tools. As the Rhapsody API evolves, so will Latitude. Registered users of CodeWarrior Latitude will receive all developer releases, the first full release, plus one additional product update, to ensure that you have access to the most up-to-date Rhapsody porting tools available.

(SCWLAT) Our Price **$399**

NEW PRODUCT!

MW Visual SourceSafe Release 5

by Metrowerks

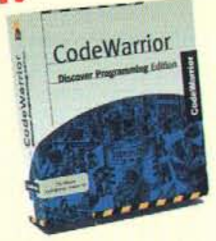


- Source code control system, plug-in to the CodeWarrior IDE
- Compatible with Microsoft Visual SourceSafe version 5.0
- Cross-platform support (Mac, Windows, UNIX)
- Configuration management in excess of 4 billion files
- Over 8,000 files and sub-projects in a single sub-project
- Registered users receive one year of free technical support and one free update

(SMWVSS) Our Price **$499**

NEW PRODUCT!

CodeWarrior Discover Programming Edition



by Metrowerks

- Learn to program in C/C++/Java/Pascal
- Full-featured programming product - not a "lite" edition
- Online books and tutorials
- Hosted on either Mac OS or Windows 95/NT

CodeWarrior Discover Programming Edition offers an unbeatable combination of full-featured programming tools, online books and instructional materials, all at a great price. Whether you want to learn the basics of programming or add a new language to your skill set, Discover Programming puts the information and tools you need at your fingertips: the award-winning, easy-to-use CodeWarrior Integrated Development Environment (IDE), four of the most popular programming languages, online books, online tutorials, sample source code and free technical support (with your registration). Discover what you can learn with CodeWarrior, buy your copy today!

(SCWDISCMAC) Mac OS Our Price **$79**
(SCWDISCWI) WIN 95/NT Our Price **$79**

Order Toll-free
800-MACDEV-1
(800-622-3381)

Apple Dylan
dynamic object oriented language and development environment

Apple Dylan Technology Release

by Apple Computer, Inc.

- Contains a PowerPC-native prototype version of a development environment based on the Object Oriented Dynamic Language (OODL) Dylan. Developers will be able to produce code targeting both 680x0 and Power Macintosh systems
- Automatic memory management
- Application framework and user-interface & builder
- High-level exception handling
- Cross-language support for C code and APIs

CD & Online Documentation
(SADTRO) Our Price **$39.95**
CD & Hardcopy
(SADTRH) Our Price **$59.95**

Newton Toolkit 1.6

by Apple Computer, Inc.

With Newton Toolkit, you can easily create software that runs on any Newton PDA, including Apple's MessagePad and Motorola's Marco.

- Now supports Newton 2.0
- Dynamic Language Eases Development
- Allows you develop applications interactively
- New Compiler Enables Faster Applications

Newton Toolkit 1.6
(SNETO) Our Price **$149**
Newton Toolkit Update 1.6
(SNETOUP) Our Price **$29**

Newton

Newton Toolkit 1.6 for Windows

by Apple Computer, Inc.

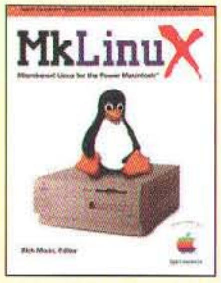
With Newton Toolkit, you can easily create software that runs on any Newton OS device from Apple or its licensees. The Newton Toolkit is a development environment that reduces the complexity and time involved in creating great software products.

- Provides access to features specific to the Newton 2.0 OS - Allows developers to create new or modify existing Newton applications to take advantage of powerful Newton 2.0 OS features
- Supports Windows 95, Windows NT 3.5x, and Windows 3.1(with Win32s) operating systems -Enables developers to use multiple development environments/operating systems when creating Newton applications
- Includes Macintosh-to-Windows NTK project converter tool - Allows developers to automate conversion of existing NTK projects from Macintosh platform to Windows platform
- Provides NS Debug Tools - Allows developers access to interactive Newton application debugging tools. Offers extended NewtonScript debugging functions, which let developers study and manipulate an application running on a Newton device. Developers can use NS Debug Tools to set break points in an application after it's been compiled and installed, step through program execution, examine and change objects on the Newton, and display a textual representation of the interpreter instructions

(SNETOW) Our Price **$149**

Lowest price guaranteed. Feel Good about your purchase! 30-day return policy.

Check out our Web site!
• Full product descriptions • Hundreds of more products
http://www.devdepot.com



NEW PRODUCT!

MkLinux: Microkernel Linux for the Power Macintosh

by Prime Time Freeware

MkLinux is a native port of Mach 3 and the Linux 2.0 kernel, complemented by hundreds of commands from BSD, GNU, and X11. It runs on most (NuBus abd OCU bus) Power Macintosh systems; Performa, PowerBook, and multiprocessor ports are currently under development. MkLinux is robust, powerful, freely distributable, and source code compatible with most other Linux systems. It provides a full suite of development tools, support for AppleTalk, HFS, and Objective-C, and access to a vast amount of free software. MkLinux is a great way to "come up to speed" on Mach, UNIX, and Rhapsody.

- MkLinux user community supports FTP and web servers, development and porting efforts, and several mailing lists
- The Apple sponsored reference release contains a wealth of introductory and reference material on Linux, Mach, NeXT, and the Power Macintosh
- Includes free 3.0 upgrade

(BMKLINUX) Our Price **$50**

Pro Fortran

by Absoft Corporation

Absoft Pro Fortran combines native F90, VAX compatible F77, and C/C++ compilers into a single, easy to use environment. All compilers are link compatible and operate through a common interface.

- Graphical debugger, browsers, array display, performance profiler, linker, MRWE application mainframe
- MIG graphics library, Absoft Create Make, several utilities, the latest version of MPW and illustrated documentation
- Whole array operations, modules, interface blocks, and user-defined types or data structures
- Dynamic memory allocation and new control constructs
- F90 is link compatible with Absoft F77, C++, MrC and CodeWarrior
- It is fully compatible with Toolbox, MPW tools, and most third-party products

(SAPROF) Our Price **$899**

Order Toll-free
800-MACDEV-1
(800-622-3381)

VIP–BASIC: Visual Interactive Programming in BASIC

by Mainstay

Now you can create full-featured, stand–alone Macintosh and Power Macintosh applications in standard BASIC code! VIP-BASIC 2.0 is the fastest way to program your Macintosh.

- Rapid application development environment with application framework, mix and match: VIP–BASIC high-level subprograms
- Import pre-existing BASIC code: automatically integrate BASIC code, export C Code for compiling: automatically convert your BASIC code to C for compilation with Metrowerks' CodeWarrior

(SVIPBASIC) Our Price **$195**

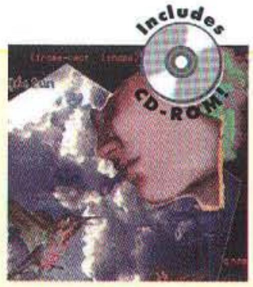
VIP–C: Visual Interactive Programming in C

by Mainstay

Now you can create full-featured, stand-alone Macintosh and Power Macintosh applications in just minutes. VIP-C 2.0 is the first rapid application development system for creating complete Macintosh programs in standard ANSI C.

- Includes powerful, tightly integrated visual debugger, Import pre-existing C code: automatically integrate C code with a current project
- Includes full-featured mini database: (ltd to 32K) of the powerful VIP–BASIC database manager gives you everything you need to setup royalty–free, multi–user database applications

(SVIPC) Our Price **$295**



Includes CD-ROM!

Macintosh Common Lisp 4.0

by Digitool, Inc.

Macintosh Common Lisp provides users with a rich set of object-oriented dynamic language features making it especially well-suited for rapid prototyping, custom development for business and education, scientific and engineering applications, and academic research.

- Power PC native environment & compiler, full Macintosh support
- CLOS, the standard Common Lisp object system
- Interactive dynamic environment, multiple processes
- Automatic memory management and self-typing data
- Ephemeral garbage collector, smaller application footprint
- Compiles with Common Lisp industry standard and smart programmable tools, 110+ mb of user contributed code
- Complete on-line documentation (manual sold separately)
- Software license and registration card

(SMCLISP) Our Price **$675**

ObjectMaster Professional Edition

NEW PRODUCT!

by Altura Software, Inc.

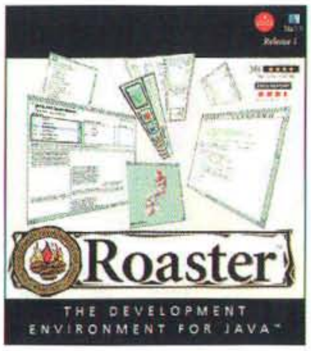
Object Master is an innovative programming environment that provides all the necessary tools to write, organize, and navigate through source code.

- Write code using the most robust source code editor available on the desktop
- Organize source code into projects to quickly access and manipulate all files
- Navigate through source code using intuitive graphical Browser windows

(SOMPE) Our Price **$399**

Roaster

by Roaster Technologies, Inc.



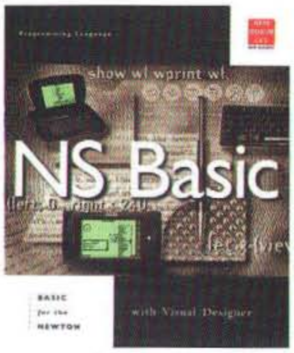
Roaster Release 3.1 is now available with JDK 1.1 and JavaBean support! Getthe most out of Sun's Java™ programming language with this powerful development environment.

- Features include: ability to build stand-alone Macintosh applications or applets; visual interface builder; ability to create cross-platform zip files; powerful Java debugger; wizard for quickly creating Java applets or applications; JavaBean support; JDBC support; Java object database included; ability to call AppleScripts from Java; Just-in-time (JIT) compiler; JDK 1.1.x support; class tree and hierarchical class browser; much more!
- Software includes: Over 300 example applets and applications; Netscape Internet Foundation Classes; Object Design's PSE for Java; OpenLink Software's JDBC Drivers; OpenSpace Java Generic Library; Microline Component Toolkit Lite 3.0; much more!
- Requirements: Runs on 68k or PowerPC, CD-ROM
- Price includes all web-based updates

(SROAST3) Our Price **$99**

NS BASIC 3.6 for the Newton with Visual Designer

by NS BASIC Corporation



- A fully interactive implementation of BASIC programming language
- Runs entirely on the Newton – no host is required
- Create files, access the built in soups, and the serial port for input and output
- Work directly on the Newton, or through a connected Mac/PC and keyboard
- Get the **BASIC Internet Tool**, available at no charge to NS BASIC users from www.nsbasic.com
- Release Notes with sample code are available from the same location
- Runs on any Newton MessagePad 130 with NS BASIC and the Newton Internet Enabler. Also runs on MP 1201s with NOS 2.0 that have full memory available
- Write short programs to access News, mail and the web

(SNSBASIC) Our Price **$99**

TENON INTERSYSTEMS

CodeBuilder

by Tenon Intersystems

CodeBuilder is a powerful and unique Macintosh software development tool for porting existing apps or developing new, advanced applications on Power Macs and Power Mac clones.

- A powerful Macintosh software development tool suite of C, C++, Objective-C, Java, Ada, and Fortran development tools
- Complete UNIX & X development environment for developing UNIX or Macintosh apps
- Includes compilers and source-code debugger for Objective C, and C, C++, Ada 95 and Fortran 77
- Web & internet scripting tools: Perl, MacPerl, tcl/tk, bash, sh, and csh
- Supports Rhapsody kernel APIs and Rhapsody TCP sockets

(SMIOCODEB) Our Price **$149**

WAIT... There's More!

Here's a list of all available products. For full product descriptions please see our Web site, or feel free to call, fax, or E-mail us.

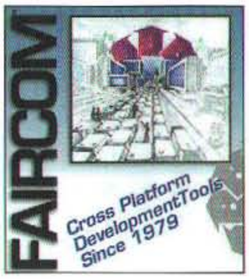
PRODUCT	CODE	OUR PRICE
LPA MacProlog Developers Edition	SLPAD	995.00
LPA MacProlog Programmers Edition	SLPAP	495.00
LS Fortran Pro	SLSFORT	595.00
LS Fortran Plug-In	SLSFPI	199.00
Mac FORTRAN II	SFORT2	595.00
Power MachTen–UNIX	SM10PPC	695.00
Presenting Magic Cap	BPRESMAGIC	15.25
SmalltalkAgents	SSTA	695.00
Think Pascal 4.0	SPASCAL	165.00

ObjectSet Mail SDK
by Smartcode Software

- Powerful C++ classes for integrating Internet e-mail in your applications
- Helps you write software that can share mail with other leading e-mail products
- Royalty-free MIME, SMTP, and POP3 APIs
- Gives you the most robust MIME parser and encoder available
- Ideal for use in Internet and Intranet environments
- Comes complete with samples with documented, reusable source code
- Free standard technical support

(SOSMSDK) Our Price **$495**

r-tree Report Generator
by Faircom



Handles virtually every aspect of report generation:

- Complete C source
- Complex multi-line reports
- Multi-file access
- Complete layout control
- Conditional page breaks
- Nested Headers and Footers
- Horizontal Repeats

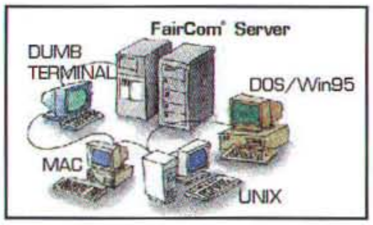
(SRTRG) Our Price **$445**

Spellswell Plus 2.1
by Working Software

- Award-winning, comprehensive, practical spelling checker that works in batch mode or within applications that incorporate the Apple Events Word Services protocol (e.g., Eudora, WordPerfect, Communicate!, and InfoDepot)
- Checks for spelling errors as well as common typos like capitalization errors, spaces before punctuation, double word errors, abbreviation errors, a/an before vowel/consonant, etc.
- MacTech orders include developer kit with Writeswell Jr., a sample AppleEvents Word Services word-processor and its source code
- Available for OEM Sales

(SSPELL) Our Price **$49**

c-tree Plus® Database Handler
by Faircom

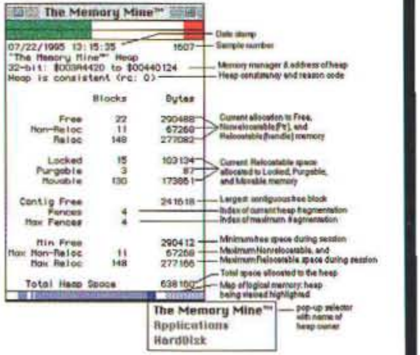


Unsurpassed Cross Platform Tools for Mac Developers!

- Full C Source
- Client/Server Option
- Over 16 years proven reliability
- Concurrent simultaneous access of Mac/PC files
- Superior throughput and performance
- Unparalleled scalability and flexibility
- Fixed/Variable length files

(SCTPDH) Our Price **$895**

Memory Mine
by Adianta Inc.


```
The Memory Mine™
07/22/1995 13:15:35                1607
"The Memory Mine™" Heap
32-bit: $003A4420 to $00440124
Heap is consistent (rc: 0)

                  Blocks      Bytes

          Free       22      290488
     Non-Reloc       11       67268
         Reloc      148      277082

        Locked       15      103134
     Purgeable        3          87
       Movable      130      173861

   Contig Free               241618
        Fences        4
    Max Fences        4

      Min Free               290412
 Max Non-Reloc       11       67268
     Max Reloc      148      277166

  Total Heap Space           638160

The Memory Mine™
Applications
HardDisk
```
Date stamp
Sample number
Memory manager & address of heap
Heap consistency and reason code
Current allocation to Free, Nonrelocatable (Ptr), and Relocatable (handle) memory
Current Relocatable space allocated to Locked, Purgable, and Movable memory
Largest contiguous free block
Index of current heap fragmentation
Index of maximum fragmentation
Minimum free space during session
Maximum Nonrelocatable, and Maximum Relocatable space during session
Total space allocated to the heap
Map of logical memory: heap being viewed highlighted
pop-up selector with name of heap owner

- Monitor heaps, identify problems such as memory leaks, and stress test applications
- Active status of memory in a heap is sampled on the fly: allocation in non-relocatable (Ptr), relocatable (Handle) and free space is shown, as are heap corruption, fragmentation, and more
- Allocate, Purge, Compact, and Zap memory lets users stress test all or part of a program

(SMEMMINE) Our Price **$99**

NEW PRODUCT!

ScriptDemon
by Royal Software, Inc.

ScriptDemon is a browser plug-in that, for the first time, allows you to deliver and run AppleScripts from Web pages. The ScriptDemon plug-in will execute the embedded AppleScript code included on a Web page. ScriptDemon painlessly and inexpensively handles many previously impossible tasks, such as:

- Using the Intranet to manage all Macintosh computers on-line
- Using the Internet to install and configure software
- Using the Internet to configure hardware
- Delivering complex sets of files and assembling them on the browsing computer
- Providing interactive education and product support on both the Internet and Intranet
- A perfect companion to LiveCard!

(SSDEMON) Our Price **$949**

STEPUP SOFTWARE

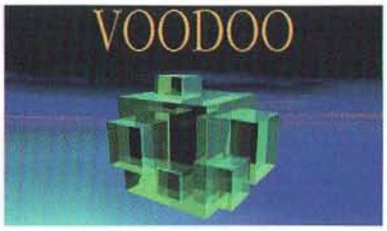
Guide Composer™ 1.2
by StepUp Software

- Create powerful Apple Guide help systems for any new or existing Macintosh application
- Provides a WYSIWYG development environment: Guide content is developed in Guide windows
- Design topics, phrases, and panels in the same format as the user will use them
- Features are WYSIWYG interface, Topics, phrases, and hierarchical phrases, Coach marks, Fully-Integrated with Apple's Guide Maker (distributed with Guide Composer), compiles scripts automatically, PICTs in Panels, Generated Guide scripts are modifiable
- FREE Update to all registered Guide Composer users. Demo is available at http://www.guideworks.com/

(SGCOMP) Our Price **$99**

SEE RELATED PRODUCTS: AppleGuide Complete, Danny Goodman's AppleGuide Starter Kit, Real World AppleGuide

VOODOO 1.8
by UNI SOFTWARE PLUS



- Stand-alone version control tool for all sorts of projects (software development, documentation, design, CAD, publishing, etc.)
- Smooth integration with Metrowerks CodeWarrior IDE
- Simple and clear management of variants and revisions of entire projects (not only of single files)
- Easy-to-use graphical project browser gives access to all versions that were ever stored.
- Recording of the complete history (who made which changes when and why)
- View differences between versions (not only for text files!)
- Efficient delta storage of arbitrary files (text as well as non-text files) gains savings of 95 % and more
- Administration of users with hierarchical access rights
- Configurable local file locking (Finder flag or 'ckid' resource)
- Scriptable, essential parts PowerPC native

Single license (SVOODOO1) **$229**
2 pack (SVOODOO2) **$359**
5 pack (SVOODOO5) **$799**
10 pack (SVOODOO10) **$1369**
20 pack (SVOODOO20) **$2399**

Additional pricing available on request.
SEE RELATED CATEGORY: Dev. Environments

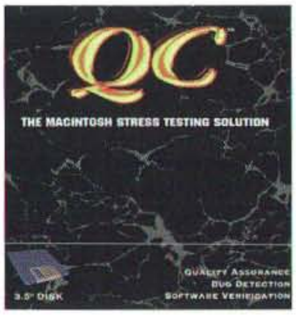
StoneTable 68K/PPC
by StoneTablet Publishing

StoneTable is a powerful and professional replacement for the List Manager used by developers worldwide. Version 3.0 is a new release with many improvements including better clipboard and drag/drop integration with other applications.

- Available for use with CodeWarrior C & Pascal
- Includes libraries for 68K (A4 & A5) and PowerPC
- An LTable-like class is provided to incorporate StoneTable into the PowerPlant environment

(SSTONEFAT) Our Price **$199**

QC
by Onyx Technology, Inc.



High performance runtime stress testing for applications.

- Tests include heap checks, purges, scrambles, handle/pointer validation, dispose/release checks, write to zero, de-reference zero as well as other tests like free memory invalidation and block bounds checking
- Extremely user friendly – ideal for non-programmer testers
- Also available in Japanese

(SQC) Our Price **$99**

Spotlight
by Onyx Technology, Inc.

NEW PRODUCT!

Spotlight is a stand alone debugging aid that performs memory protection (arrays, heap accesses, outside your heap, low mem, etc), discipline checking on toolbox calls, and leaks detection.

- Spotlight is sold on an annual subscription basis
- The subscription service provides all updates
- Includes maintenance releases for one year after the initial purchase or renewal date.

(SSPTLT) Our Price **$199**

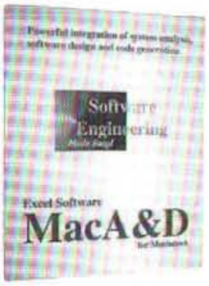
AppSketcher 1.0 for BeOS
by BeatWare, Inc.

AppSketcher is the premier programming tool for the BeOS.

- The fastest way to develop software on the BeOS
- Drag and drop design is great for creating user interfaces
- Supports localization for worldwide sales
- Expands applications easily without manual code modifications
- Shortens development time by automating the Make process through direct communication with the BeIDE

(SASFB) Our Price **$199**

MacA&D 6.0
by Excel Software



- Structured analysis and design
- Object-oriented analysis and design
- Real-time and multi-task design
- Data and screen modeling
- Integrated code editing and browsing
- Multi-user dictionary and requirements
- Code to design diagrams for C, C++,etc.
- Design diagrams to code for C, C++, etc.
- State modeling diagrams and tables
- Use cases with traceability

(SMACADP) Our Price **$1995**

Order Toll-free
800-MACDEV-1
(800-622-3381)

Apprentice 6
by Celestin Company

Apprentice 6 is a high-quality CD-ROM collection of over 600 megabytes of up-to-date source code, utilities, and info for Mac programmers. All of the source code and utilities are completely new or updated for this release.

- Frontier 4.1, the highly-acclaimed scripting environment
- More PowerPlant AND many more PowerPC samples
- Cool new languages and environments added (Clean, Eiffel, F, Tcl-Tk)
- Hot new demos from leading Mac development companies

(SAPPRENT6) Our Price **$35**

SoftPolish CD-ROM
by Bare Bones Software

- The essential tool for software quality assurance on the Macintosh
- Helps you identify inconsistencies with Apple's user interface guidelines, misspelled words, missing resources, and other mistakes
- Provides tools to put the finishing touches on software distribution packages prior to release
- Works independently of any programming language or environment
- Ideal for sanity checking software throughout the development process

(SSOFTPOL) Our price **$99**

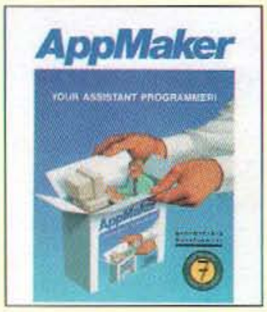
Future Basic II
by Staz Software

FutureBASIC II is the award winning leader in Macintosh BASIC programming.

- Source level debugger and Interactive compiler/editor
- Multi-file Project manager and Multi-file find and replace
- Super fast compilation, 32 bit clean, and System 7.x savvy
- QuickBASIC converter
- Getting Started manual with over 500 example files
- Full support of standard BASIC

(SFBASIC2) Our Price **$229**

AppMaker
by Bowers Development



- Develop the user interface for a Macintosh application using the original interface builder
- Just point and click to design your application
- Creates resources and generates excellent source code
- Supports most development environments including Metrowerks, Symantec, or MPW; C, C++, or Pascal; procedural or object-oriented, using PowerPlant, TCL, or MacApp
- The generated code uses the Universal Headers to provide PowerMac compatibility
- Great tool for beginners to learn object-oriented and Macintosh Toolbox programming techniques
- Includes one-year subscription on CD and hardcopy documentation

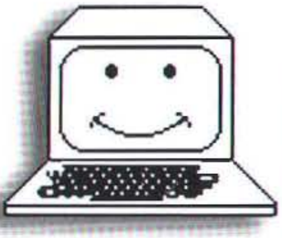
(SAPPMAKE) Our Price **$299**

dtF
by dtF Americas

- True relational database system for Apple Macintosh computers
- Provides a powerful choice for developers who want to create database centered applications with no performance trade-offs
- Features SQL, full transaction control, error recovery, single user, client server architecture and multi-platform support including DOS, Windows, OS/2 and UNIX
- The C/C++ API is identical and fully portable across all supported platforms
- Third-party vendors supporting dtF will be able to offer a variety of advanced features and benefits to their customers royalty free
- Tools are included for importing, exporting, creating and managing databases and users
- Supported development environments include: Symantec, MPW, Metrowerks and more Mac/SDK

(SDTF) Our Price **$695**

B-Tree HELPER 2.2
by Magreeable Software



- Inexpensive database engine for Macintosh programmers in C source code
- Uses contiguous fixed length blocks
- Expands the file as necessary and contracts files when possible
- Inserts and deletes keys in one or more B-Trees
- Finds keys equal to, less than, or greater than a given value in a few hundredths of a second
- Finds lists of records whose keys are equal to, less than, or greater than a given value or are in a range of values

(SBTREE) Our Price **$149**

Step-Up Installer Pack
by StepUp Software

- Package of several Installer "atoms" that let developers incorporate graphics, sounds, file compression and custom folder icons into installation scripts
- Compression formats supported are Compact Pro & Diamond
- Each atom also available separately
- Compression requires additional licensing

(SINSTALL) Our Price **$219**

ScriptGen Pro
by StepUp Software

- Installer script generator which requires no programming or knowledge of Rez
- Supports StepUp's InstallerPack, StuffIt decompression, Compact Pro decompression, custom packages, splash screens, network installs, and resource installation

(SSCRPTGEN) Our Price **$169**

Water's Edge Software

Tools Plus libraries + framework

by Water's Edge Software

Easily create compact, fast running, professional looking applications and plug-ins*. Tools Plus lets you create virtually any user interface element with a single routine, and it transparently provides a robust infrastructure to make all your pieces work together as an application. Rated 4 stars by Macworld! MacTech (July 96): "it's an incredibly rich collection of tools... If you are interested in developing applications that have 'quality' written all over them, then Tools Plus is for you."

- Simplifies programming and thins source code
- Automates all standard GUI elements
- Thousands of extras, from floating palettes and tool bars to powerful picture buttons
- Includes numerous 3D grayscale options
- Over 1/2 MB of custom fonts, icons, cursors, and other resources
- Includes SuperCDEFs world-class controls (an $89 value) free

(STOOLCW) Our Price **$249**
CodeWarrior Gold (C/C++ & Pascal, 68K & PPC)
(STOOLCWB) Our Price **$199**
CodeWarrior Bronze (C/C++ & Pascal, 68K)
(STOOLSYMT) Our Price **$199**
Symantec (THINK) C/C++ and THINK Pascal (68K)
(STOOLSYM) Our Price **$149**
Symantec (THINK) C/C++ (68K)
(STOOLPAS) Our Price **$149**
THINK Pascal (68K)

*CodeWarrior required to write plug-ins

Order Toll-free
800-MACDEV-1
(800-622-3381)

TestTrack-Bug Tracking the Macintosh Way

by Seapine Software, Inc.

- Tracks bugs, feature requests, test configurations, users, and more
- Includes notifications, security, a powerful filter mechanism, and multiple reports
- Links your testers, engineers, documentations staff, and project managers together to ensure all bugs are identified, fixed, and documented
- Eliminates the need to build custom bug tracking solutions using general purpose database tools
- Supports single- and multi-user bug databases (additional licenses required to use multi-user features)

(STETR) Our Price **$129**

CodeBuilder

by Tenon Intersystems

TENON INTERSYSTEMS

CodeBuilder is a powerful and unique Macintosh software development tool for porting existing apps or developing new, advanced applications on Power Macs and Power Mac clones.

- A powerful Macintosh software development tool suite of C, C++, Objective-C, Java, Ada, and Fortran development tools.
- Complete UNIX & X development environment for developing UNIX or Macintosh apps
- Includes compilers and source-code debugger for Objective C, and C, C++, Ada 95 and Fortran 77
- Web & internet scripting tools: Perl, MacPerl, tcl/tk, bash, sh, and csh
- Supports Rhapsody kernel APIs and Rhapsody TCP sockets

(SMIOCODEB) Our Price **$149**

Phyla™: Object–Oriented Database

by Mainstay

- Powerful Databases Without Programming: Phyla handles your all your complex database needs
- Define a Database in Minutes: Using an intuitive, graphical user interface
- Objects Are a More Natural Approach: Phyla creates real world databases
- Drag–and–Drop Ease: Relate objects by simply dragging objects between windows
- Create Custom Forms and Reports: Quickly create custom forms and reports
- Fast Finds and Sorts: Perform complex queries and calculations without programming
- Synchronize Multiple Databases Copies
- Password Protection With Access Limitations
- Easy Import and Export: Import from other databases, export data in various formats

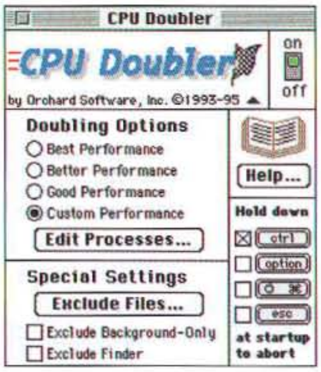
(SPHYLA) Our Price **$179**

BBEdit 4.0.4

by Bare Bones Software

A powerful, easy-to-learn text editor. Adds new features for HTML coders, including a spelling checker and HTML tag palette. Accelerated for Power Macintosh; dragging supported everywhere; Internet Config aware; PowerTalk aware. Integrated support for Symantec's IDE, Metrowerks CodeWarrior, THINK Reference 2.x, MPW ToolServer, and most other environments. Many UNIX style tools, including "grep" searches, file comparisons, and sorting multi-file search and replace. PopUpFuncs feature lets you jump to a function from a menu.

(SBBEDIT) Our Price **$119**


CPU Doubler
CPU Doubler
by Orchard Software, Inc. ©1993-95
on off
Doubling Options
○ Best Performance
○ Better Performance
○ Good Performance
◉ Custom Performance
Edit Processes...
Help...
Hold down
☒ ctrl
☐ option
☐ ⌘
☐ esc
at startup to abort
Special Settings
Exclude Files...
☐ Exclude Background-Only
☐ Exclude Finder

CPU Doubler
by Orchard Software

- Performance enhancement utility for the Macintosh
- Increases the speed of your computer by 100%
- Works on both the PowerPC and 68K Macintosh
- Manages computer throughput using a proprietary scheduling algorithm
- Ensure optimal performance and compatibility

(SCPU2X) Our Price **$79**

SEE RELATED PRODUCTS:
Development Environments

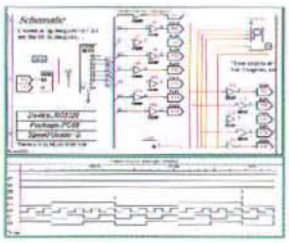
CompileIt!
by Royal Software, Inc.

NEW PRODUCT!

CompileIt!, the first HyperTalk compiler, is a complete developement system for the creation of XCMDs and XFCNs.

- Expand the capabilities of your environment by using CompileIt! and the ROM Toolbox extensions
- Increase the speed of routines written in HyperTalk by turning scripts into externals
- Protect sensative code from prying eyes because your code is now compiled!
- Easily learn Macintosh programming by exploring the ROM Toolbox
- Includes DebugIt!, a valuable source-level debugger for externals created with CompileIt!

(SCOMPIT) Our Price **$149**



DesignWorks 4.0
by Capilano Computing

DesignWorks 4.0 has all the ease of use and schematic editing power of previous versions, plus new features designed to make your entire design process easier and more error free. The 4.0 version has new menu customization and scripting features that will directly address your design checking and interfacing needs.

- Flexible schematic editing features speed the drawing process
- Full Undo/Redo on all editing operations
- Hierarchical design with unlimited levels is fully supported
- Powerful attribute features allow arbitrary text information to be associated with any signal, device or device pin
- Extensive symbol libraries with over 12,000 parts in ANSI and IEEE format
- Integrated device symbol editor allows you to create custom symbols using standard drawing tools
- Interactive digital simulator option is available. No netlists, no application switching!

(SDWORKS) Our Price **$995**

EtherPeek
by AG Group, Inc.

NEW PRODUCT!

EtherPeek for Macintosh is a full-featured protocol analyzer that allows you to quickly and easily test and debug network communication, and:

- Check for protocol compliance
- Use hundreds of built-in decodes
- Develop custom packet decoders
- Filter packets during or after capture
- Test device reactions to specific packet types
- Customize or alter packets for transmission
- Generate traffic to test varying loads

(SEPEEK) Our Price **$745**

OpenGL for the Macintosh
by Conix Graphics

OpenGL

OpenGL is the premier 3D graphics library that allows software developers the ability to develop high-quality, interactive 2D and 3D graphics applications. OpenGL can perform the following wide range of functions which will enhance the development of all graphics software:

- Geometric primitives (points, lines, and polygons)
- RGBA or color index mode
- Viewing and modeling transformations
- Texture Mapping, Lighting, Shading and Z Buffering
- Atmospheric Effects (fog, smoke, and haze)
- Alpha Blending (transparency)
- Antialiasing, Accumulation Buffer, Stencil Planes
- Display list or immediate mode
- Polynomial Evaluators (to support Non-uniform rational B-splines)
- Feedback, Selection, and Picking Raster primitives (bitmaps and pixel rectangles)
- Pixel Operations (storing, transforming, mapping, zooming)

(SOPENGL) Our Price **$389**

VText
by Vivistar

VText is a C++ add-on library for Metrowerks' PowerPlant application framework. VText provides complete Macintosh text support including: greater than 32kb text, undo, drag and drop editing, AppleEvent scripting and recordability, full support for multibyte characters and inline input methods including Japanese and Chinese text, and full support for bi-directional script systems including Arabic and Hebrew.

- Full featured text engine for Metrowerks' PowerPlant
- Stylesets and rulersets with tabs
- Flexible object oriented C++ API
- Full undo and drag and drop editing
- WorldScript savvy including bidirectional and multibyte scripts with inline editing
- AppleEvent factored for scriptability and recordability

(SVTEXT) Our Price **$349**

QUED/M 3.0

by Nisus Software

- The programmer's text editor that defined the industry standard for speed and efficiency
- PowerPC native
- Features integrated support for Symantec C/C++, Metrowerks CodeWarrior 6, and MPW
- Supports all the major development environments on the Macintosh.
- Powerful editing features, including unlimited undo and redo, macro language, scripting, text folding, ten editable/appendable clipboards, markers, displaying text as ASCII codes, dynamic coloring of C/C++ keywords/comments, rectangular and non-contiguous selection
- Includes Celestin Company's APPRENTICE 4

(SQUEDM) Our Price **$89**

AG Author

by Lakewood Software

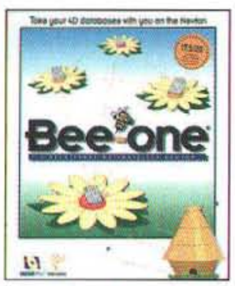
AG Author 1.0 is a full-featured Apple Guide authoring tool with fully customizable project template. The following features are unique to AG Author:

- Support for styled, colored, & hot text
- Fully customizable project template
- Flexible compile options
- Find & replace tool for scripts
- Multiple open projects
- Rapid deployment of project globals

NEW PRODUCT!

(SAGA) Our Price **$99**

SEE RELATED PRODUCTS: AppleGuide Complete, Danny Goodman's AppleGuide Starter Kit, Real World AppleGuide



Bee-one

by Power Box

Bee-one lightens your load on the road by adapting relational databases developed under 4D© to the Newton Platform. Once the program is installed on both the Macintosh and the Newton, it takes 4 simple steps to use Bee-one!

- Database transfer, Set-up, Use, and Synchronization

(SBEEONE) Our Price **$139**

Web Ware

by BeachWare, Inc.

The ultimate collection of clip media and templates for building your own Web Page. An incredible selection of Shockwave movies, animated GIFs, buttons, bullets, dividers, and sample HTML pages. There are literally thousands of graphical elements on this disc, all there to spice up your web page. In all, it's about 300 megabytes of creativity only a mouse-click away! System Requirements: PC - 486 or better with 8 MB RAM, Sound card, SuperVGA, CD-ROM drive. Macintosh - Color Mac with 8 MB RAM, CD-ROM drive.

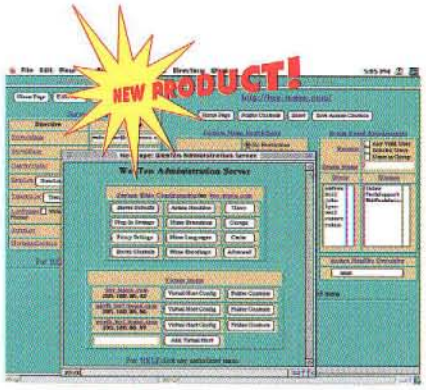
(SWEBW) Our Price **$24**

WAIT... There's More!

Here are more products. For full product descriptions please see our Web site, or feel free to call, fax, or E-mail us.

PRODUCT	CODE	OUR PRICE
Fortran 77 SDK	SF77	699.00
ICONIX PowerTools-10 Pack	SICPP10	7,845.00
ICONIX PowerTools-6 Pack	SICPP6	5,945.00
ICONIX PowerTools-8 Pack	SICPP8	6,945.00
ICONIX PowerTools-AdaFlow	SICADA	1,395.00
ICONIX PowerTools-ASCII Bridge	SICASCII	1,395.00
ICONIX PowerTools-CoCoPro	SICCOCO	1,395.00
ICONIX PowerTools-DataModeler	SICDATAMOD	1,395.00
ICONIX PowerTools-FastTask	SICFASTTASK	1,395.00
ICONIX PowerTools-FreeFlow	SICFREEFL	1,395.00
ICONIX PowerTools-Object Modeler	SICOBJMOD	1,395.00
ICONIX PowerTools-PowerPDL	SICPOWER	1,395.00
ICONIX PowerTools-QuickChart	SICQUICKCH	1,395.00
ICONIX PowerTools-SmartChart	SICSMART	1,395.00
ICONIX Training & Consulting	TICONIX	2,945.00
IMSL Math and Stat Library	SIMSLSTAT	495.00
Info-Mac X	SINFOMAC10	39.95
Ionizer Real-Time Spectral Reshaping Tool	SIONIZER	800.00
LiveAccess™ 1 User Edition	SLAUE	69.00
LiveAccess™ 1 Developer Edition	SLADE	99.00
LiveCard	SLCARD	149.00
LJ Profiler	SLJPROF	295.00
MacFlow™: Flowchart Design and Development	SMACFLO	179.00
Mac Source II	SMACSOURCE	29.95
Nisus Writer 5.0	SNISUSW	220.00
Plan & Track™: Project Planning and Management	SPLNTRK	179.00
Screen Machine	SSM	24.00
Spyer	SSPY	39.00
Visual Café	SVCAFEMAC	199.00

WebTen
by Tenon Intersystems



WebTen is an industrial-strength, high-performance Apache Web server for Power Macs. WebTen's Web-based browser interface enables local or remote administration via your favorite browser. Since Apple's NeXT acquistion, Tenon has extended their unique "UNIX virtual machine" technology to produce a set of "Rhapsody-Ready" internet applications. WebTen is the first offering in this series.

- WebTen is the fastest Web server on Power Macintosh
- Sustains up to 10,000 connections a minute, or over 10 million connections a day
- Apache runs in Tenon's multi-threaded, pre-emptive multitasking environment
- Tenon's unique technology supports the widely acclaimed Apache Web server as a double-clickable Macintosh application (SWEBTEN) Our Price **$495**

OOFILE Reporter Writer
by A.D. Software

- Full embedded report-writer, allows you to preview page-by-page and either print or save as plain text, HTML or RTF
- Multiple levels of breaks, database views, headers and footers are provided using a clean object-oriented design
- Incudes RAM-based version of OOFILE database. Included in full OOFILE Platform Bundle
- Saving to file without preview of printing is cross-platform-run on your Mac/Win/Unix server and creates web pages
- Price includes 1-year subscription (SOORW) Our Price **$499**

NEW PRODUCT!

WebSiphon
by Purity Software, Inc.

WebSiphon is one of the most anticipated new CGI products for Macintosh Webmasters delivering a complete authoring tool for truly revolutionary sites. This product alone can replace most all CGIs on your site resulting in increased dynamic serving speed, reliability, and powerful scripting and database abilities directly within your HTML pages! Also includes Verona, the fastest flat-file database server available for Macintosh web sites.

(SWSIPHON) Our Price **$495**

PageCharmer 1.0
by Mainstay

PageCharmer is a set of customizable interactive applets that enhance web pages without writing a single line of HTML code. Whether the web site is already up and running or designing one from scratch, PageCharmer gives you the power to make it stand out from the crowd with sophisticated applets that can be personalized to fit most any need.

FEATURES:
LiveG-Map, LiveT-Map, LiveG-Button, LiveT-Button, LiveGT-Button, LiveG-Ticker, LiveT-Ticker, LiveG-Marquee, and LiveT-Marquee.
(SPGCHRM) Our Price **$139**

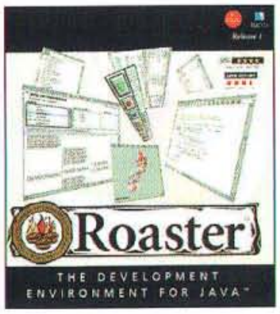
BBEdit 4.0.4
by Bare Bones Software

A powerful, easy-to-learn text editor. Adds new features for HTML coders, including a spelling checker and HTML tag palette. Accelerated for Power Macintosh; dragging supported everywhere; Internet Config aware; PowerTalk aware. Integrated support for Symantec's IDE, Metrowerks CodeWarrior, THINK Reference 2.x, MPW ToolServer, and most other environments. Many UNIX style tools, including "grep" searches, file comparisons, and sorting multi-file search and replace. PopUpFuncs feature lets you jump to a function from a menu.

(SBBEDIT) Our Price **$119**

ObjectSet Mail SDK
by Smartcode Software

- Powerful C++ classes for integrating Internet e-mail in your applications
- Helps you write software that can share mail with other leading e-mail products
- Royalty-free MIME, SMTP, and POP3 APIs for Macintosh, Windows, and Unix
- Gives you the most robust MIME parser and encoder available
- Ideal for use in Internet and Intranet environments
- Comes complete with samples with documented, reusable source code
- Free standard technical support (SOSMSDK) Our Price **$495**



Order Toll-free
800-MACDEV-1
(800-622-3381)

Roaster
by Roaster Technologies, Inc.

Roaster Release 3.1 is now available with JDK 1.1 and JavaBean support! Get the most out of Sun's Java™ programming language with this powerful development environment.

- Features include: ability to build stand-alone Macintosh applications or applets; visual interface builder; ability to create cross-platform zip files; powerful Java debugger; wizard for quickly creating Java applets or applications; JavaBean support; JDBC support; Java object database included; ability to call AppleScripts from Java; Just-in-time (JIT) compiler; JDK 1.1.x support; class tree and hierarchical class browser; much more!
- Software includes: Over 300 example applets and applications; Netscape Internet Foundation Classes; Object Design's PSE for Java; OpenLink Software's JDBC Drivers; OpenSpace Java Generic Library; Microline Component Toolkit Lite 3.0; much more!
- Requirements: Runs on 68k or PowerPC, CD-ROM
- Price includes all web-based updates

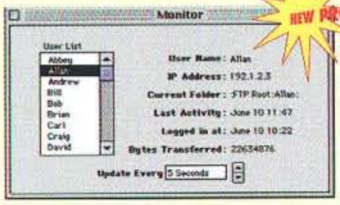
(SROAST3) Our Price **$99**

Power MachTen 4.0.3
by Tenon Intersystems

MachTen is the only Macintosh product that can turn your Macintosh into a complete Unix workstation. Based on BSD4.4 and the Mach kernel, MachTen brings the power of Unix to your desktop at an extremely attractive price point. MachTen enables you to:

- Run a high speed internet server, complete with WWW, FTP, NFS, DNS and print service
- Build a Mutihomed Web Server
- Develop applications in a Unix development environment, replete with the acclaimed GNU development toolset
- Program in Ada, C, C++, Pascal, Fortran, and more
- Run Xwindows applications, from remote workstations or on your Macintosh
- Run hundreds of Unix applications, already ported for MachTen and available on our Ported Applications CD-ROM
- Run Software.com Inc's acclaimed Post.Office mail transport service

(SM10PPC) Our Price **$695**



User Name: Allan
IP Address: 192.1.2.3
Current Folder: :FTP Root:Allan:
Last Activity: June 10 11:47
Logged in at: June 10 10:22
Bytes Transferred: 22634876
Update Every 5 Seconds

NEW PRODUCT!

Rumpus
by Maxum Development

Maxum's new, high-performance FTP server for the MacOS. Based on Maxum's RushHour TCP/IP implementation, Rumpus 1.0.1 offers the performance and reliability of high-end workstations with the ease of use, security, and flexibility of the Macintosh.

- Simplified setup, with no need to configure AppleShare, File Sharing, or Users & Groups for simple anonymous FTP
- Anonymous and/or secure server access, with separate security settings for anonymous vs. secure users
- Automatic MacBinary and Binhex encoding
- Complete logging, with separate anonymous and secure access logs, including anonymous user passwords
- Up to 32 simultaneous connections

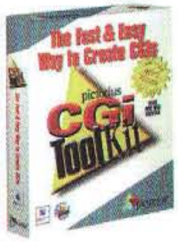
(SRUMP) Our Price **$195**

ScriptDemon
by Royal Software, Inc.

ScriptDemon is a browser plug-in that, for the first time, allows you to deliver and run AppleScripts from Web pages. The ScriptDemon plug-in will execute the embedded AppleScript code included on a Web page. ScriptDemon painlessly and inexpensively handles many previously impossible tasks, such as:

- Using the Intranet to manage all Macintosh computers on-line
- Using the Internet to install and configure software
- Using the Internet to configure hardware
- Delivering complex sets of files and assembling them on the browsing computer
- Providing interactive education and product support on both the Internet and Intranet
- A perfect companion to LiveCard!

(SSDEMON) Our Price **$949**

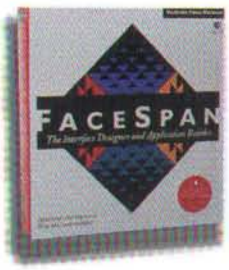


CGi Toolkit
by Pictorius Inc.

The Pictorius CGi Toolkit is the fast and easy route to high performance CGIs and ACGIs for your Mac Web site.

- Interactively develop CGIs while the web server, the CGI Toolkit and the browser are running on the same machine
- Interactively develop, test and debug CGIs before compiling
- Powerful debugger allows you to edit code, roll back, code and change input values while your application is running
- Fully object oriented so you can re-use your code
- Automatic handling of Apple Events so you can concentrate on building functionality
- Easy creation of multi-function CGIs which reduces application footprint and RAM usage

(SCGITLKT) Our Price **$149**



FaceSpan v2.1

by Digital Technology International

- Develop integrated software, make stand alone applications, create friendly interfaces
- Develop quick prototypes, print multiple pages with sophisticated layouts
- Script essential elements of the FaceSpan application - Enhanced save options
- Play and record sounds as either "snd" resources or as "AIFF" files
- Create miniature or complete apps that run on either Power PC or 68k computers
- Use precise time measurement for implementing timed behaviors — New properties
- Proportionally scale PICT images -Align images in a pictbox - Automate any application
- Monitor and respond to low-memory situations-Increased support for Frontier UserTalk!

(SFACESPAN) Our Price **$299**

PreFab Player

by PreFab Software, Inc.

PreFab Player is a faceless background application (similar to a system extension) that lets your scripts query and control otherwise non-scriptable applications, desk accessories and control panels. Player adds verbs that directly manipulate the Macintosh user interface: choose from menus & pop-ups, select radio buttons, type text, determine the name of the frontmost window, the state of a check box, etc. Balloon help identifies non-standard dialog items.

- Adds verbs to AppleScript and to Frontiers UserTalk
- Controls the Frontmost Application
- Balloons Identify User Interface Objects

(SPLAYER) Our Price **$95**

Scripter 2.0

by Main Event Software

For professionals, for novices, for webmasters, for solutions providers, there's only one serious choice. Scripter!

- Scripter and FaceSpan work together: one click opens your FaceSpan script in Scripter, another sends it back
- Debug handlers without modifying your scripts using the Call Box
- Applet simulation, live editing, Object map, associated terminology
- Search backwards, block generators, more navigation shortcuts, more drad-and-drop, and an even more enhanced trace log
- Now Includes ScriptBase; stores your data and media elements and share them between scripts all with a special new browser
- Easily write and compile scripts that have handler declarations and other vocabulary specific to a particular scriptable application
- Scripter is the natural companion to AppleScript for users at all levels of proficiency. Don't write scripts without it!

(SSCRIPTER) Our Price **$199**

AppleScript Software Development Toolkit 1.1

by Apple Computer, Inc.

- AppleScript language, system software extension, and script editor
- FaceSpan 1.0
- Developer's redistribution license for AppleScript System software extension and FaceSpan runtime code

(SASDT) Our Price **$49**

Script Debugger

by Late Night Software Ltd.

- A powerful and flexible AppleScript authoring tool – get the most from AppleScript!
- Advanced debugging environment offers single-step script execution with breakpoints
- Script Debugger dictionary browser features a graphical view of objects provided by scriptable applications
- Includes Late Night Software Scripting Additions – a collection of more than 70 new AppleScript commands, and Scheduler, a utility that allows you to launch scripts at pre-determined times

(SDEBUG) Our Price **$129**

TCP/IP Scripting Addition

by Mango Tree Software

- Award-winning AppleScript scripting addition
- Allows you to write scripts using MacTCP™ commands in AppleScript™
- Send e-mail or files through a script, check if users are logged on (via Finger), automate FTP, Gopher, NetNews, Telnet, and LPR, verify links in HTML documents, and quickly write many other TCP/IP client-server programs
- Works with AppleScript, MacTCP 2.0.4 and Open Transport

(STCP) Our Price **$49**

WindowScript

NEW PRODUCT!

by Royal Software, Inc.

WindowScript is the ultimate tool for designing Macintosh user interfaces using HyperCard. Design Real "Macintosh" user-interfaces right inside HyperCard. Until now you either created HyperCard stacks or Macintosh applications. With WindowScript you can literally bring the look and feel of a real Macintosh user-interface to HyperCard. If you're a HyperCard developer, interface designer, application developer, program manager or tester searching for a prototyping tool, WindowScript is perfect for the job.

(SWSCRIPT) Our Price **$149**

Apple Media Tool Programming Environment 2.1
by Apple Computer, Inc.

- This object-oriented language and application framework allows programmers to customize features used within the Apple Media Tool authoring environment
- Includes an expanded Apple Media Language (AML) class library, incremental compiling and linking of AML code, faster debugging facilities, Macintosh Programmers' Workshop (MPW), and user-oriented documentation written from an AMTPE developer's perspective
- Portable across 68K, Power Macintosh, and Windows platforms

(SAMTPE) Our Price **$995**

Multimedia Authoring with Apple Media Tool
by Apple Computer, Inc.

Apple Media Tool offers new multimedia users a way to get started creating interactive multimedia with minimal learning time. This self-paced tutorial will make Apple Media Tool (AMT) even easier to understand and to use. Using this tutorial, you will create a realistic multimedia project using exciting techniques such as QuickTime movies, animation and more. A demo version of AMT is included and can be used for the exercises. Training Format: Tutorial with labs.

(SMWAMT) Our Price **$49.95**

Virtual Reality Programming with QuickTime VR 2.0
by Apple Computer, Inc.

- Virtual Reality Programming Book/CD-ROM for QuickTime VR 2.0
- Enables you to write C and C++ programs using QuickTime VR 2.0
- Allows QuickTime VR to be used in games, multimedia titles and other programs
- QuickTime VR 2.0 objects can be zoomed in on, panned, or linked with hots spots
- Both panoramas and objects have hot spots linked to World Wide Web URLs

(SVRPQT) Our Price **$49**

QuickTime Developer's Kit 2.0
by Apple Computer, Inc.

- QuickTime 2.0 Extension, QuickTime Power Macintosh Extension, and QuickTime Musical Instruments extension
- Utilities like MoviePlayer 2.0, 16-bit Audio Compression, etc.
- Sample content such as MPEG Movies, Music Movies, Time-Code Movies, and 60 field per second movies
- Includes software-only playback features such as faster 2x playback mode for current compressors, Apple Cinepak compressor, 1-bit fast dithering, network tuning, load-into-RAM option, and Photo CD support

(SQTDK) Our Price **$99**

Clip VR™
by eVox Productions

Clip VR™ is a new digital image library offering high quality Photographic Virtual Reality (PVR) images for use with Quicktime® VR and other desktop VR tools.

Clip VR™ Panoramic Image components include alpha channel masks. Combine elements into a composite panorama which is converted to the finished QuickTime VR movie using Make QTVR Panorama Tool. The Components ("Clips") include complete 360 degree scenes, libraries of 360 degree terrains, 360 degree skies, buildings, and objects. Images are provided as .PICT files AND QuickTime VR movie files. Clip VR™ allows you to create high quality VR worlds from pre-photographed component images. Clip VR™ can be used to add excitement to a web site, to increase the interactive value of a CD-ROM, or simply for fun! Requires imaging editing program such as Adobe Photoshop™ to perform .PICT image compositing.

(SCLIPVR) Our Price **$89.95**

QuickTime VR 2.0 Authoring Tools Suite
by Apple Computer, Inc.

- QuickTime VR is a cross-platform software from Apple which enables webpage designers and professional developers to create new multimedia products and webpages incorporating QuickTime VR content. With QuickTime VR, users interactively navigate through 360° views of space, and explore three dimensional objects on Macintosh or Windows-based personal computers
- The QuickTime VR Authoring Tools Suite is a set of Macintosh tools to create and link panoramas and objects from photographic, digital, video, or computer generated images
- Included is a complete set of documentation for planning, designing, photographing, and creating QuickTime VR panoramas and objects. The authoring tools also allow you to link objects to panoramas using clickable hot spots

Included on the CDs are:

- A software tool (MPW-based) that stitches and blends adjacent images into a panoramic PICT file
- A software tool (MPW-based) that dices and compresses panoramic PICT files to less than 100 KB (low resolution) per panorama
- A scene editor (HyperCard-based) to create QuickTime VR scenes by adding and positioning nodes, hot spots, linking nodes together, and for linking QuickTime VR objects to scenes
- A variety of utility tools for formatting the data into the runtime software

Due to a revolutionary distortion-correcting algorithm, QuickTime VR panoramas and objects maintain a normal perspective when the user moves the mouse. The speed of the algorithm allows up to 24-bit color images. Both vertical and horizontal panning can occur at fast speeds.

(SQTVRATS) Our Price **$395**

NEW PRODUCT!

Media Cleaner Pro
by Terran Interactive

Use Media Cleaner Pro 2.0 to optimize and compress video for CD-ROM, kiosk, or the Internet. Media Cleaner Pro automates your work flow allowing you to get the highest quality video, faster and easier than any other program on the market.

- Includes Adobe Premiere Export module
- Optimal palette generation, Drag-and-drop batch processing
- RealMedia, VDOLive and improved QuickTime support
- Dynamic Preview Window, the Media Wizard, multiprocessor support and more!

System Requirements:
68040 Mac or better (PowerPC strongly recommended, req'd for RealMedia), QuickTime 2.0 or later (2.5 strongly recommended) 8 Mb application RAM, MacOS 7.0.1 (7.5 or later recommended) SoundManager 3.2, CD-ROM Drive

(SMCP) Our Price **$359**

Registered owners of Movie Cleaner Pro 1.3 or earlier can upgrade

(SMCPUP) Our Price **$129**

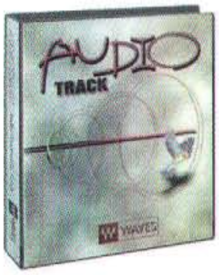
Captivate 4.6: Essential Graphics Utilities
by Mainstay

Captivate™ 4.6 is a powerful collection of graphics utilities for Macintosh, based on Mainstay's acclaimed screen capture utility, graphics and multimedia scrapbook, and graphics viewer. Captivate provides essential graphics utilities to the professional and hobbyist alike.

- Package includes: Captivate Select, Captivate View, and Captivate Store
- Any one of the three can be used alone, and together they make an unbeatable team
- Whether writing a training manual, creating an ad, or just creating a startup screen from your favorite picture, Captivate is everything professionals need at a price anyone can afford.

(SCAPTIV) Our Price **$79**

Lowest price guaranteed. Feel Good about your purchase! 30-day return policy.



AudioTrack
by WAVES

AudioTrack is a software plug-in for native processing on digital audio recording and editing systems. Waves AudioTrack combines the most-needed audio processors into a single piece of software, including 4 bands of equalization, compression/expansion, and noise gating. AudioTrack is ideal for preparing audio for InterNet streaming formats, processing individual mono/stereo tracks of audio and audio for video editing systems including digital sequencers.

Feature list

- A single window interface
- 4-band ParaGraphic Equalizer, Compressor/Expander and Gate
- Instantaneous A/B comparisons of on-line settings
- Pretested setup libraries supplied for various processing solutions
- Power Macintosh native processing (requiring no DSP board)
- Volume and gain reduction meters
- Peak hold and clip meters

Requirements:
The AudioTrack is compatible with all machines supported by Deck II, SoundEdit 16, Adobe Premiere 4.0 and Cubase VST 3.1

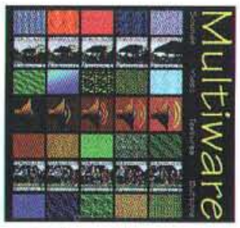
(SAUDIOTRK) Our Price **$270**

Music Tracks
by BeachWare, Inc.

A new PC/Mac & Audio multimedia music CD-ROM. The clips include musical introductions, fanfares, background music, and more. This collection offers you 100 music clips stored in .WAV format for Windows, SoundEdit & AIFF formats for Macintosh and as Audio tracks for audio CS players. All of the music clips are completely license and royalty-free!! Mac System requirements: Mac Plus or greater, CD-ROM drive. PC system requirements: Windows 3.1 or later, Sound Blaster compatible board, CD-ROM drive.

(SMT) Our Price **$24**

Order Toll-free
800-MACDEV-1
(800-622-3381)



MultiWare Multimedia Collection
by BeachWare, Inc.

Introducing a new Audio multimedia music CD-ROM for the Macintosh. This disc is a collection of clips ideal for Desktop Presentations and other Multimedia applications. This incredible collection of license-free media clips is bursting with 240+ color pictures and backdrops (PICT), 200+ sound & music clips (SoundEdit), 140+ QuickTime movies, and a variety of multimedia tools for use with the Macintosh.

(SMWMC) Our Price **$24**

Casino!

by BeachWare, Inc.

Can't make it to Vegas this month? Your best bet is Casino! Whether your favorite is Slots, Poker, Blackjack, or Keno, this virtual casino will entertain you for hours with its ten different machines. Mac System requirements: Color Mac, CD-ROM drive, 4 MB of RAM. PC system requirements: Windows 3.1 or later, CD-ROM drive, 4 MB of RAM.

(SCAS) Our Price **$24**

Abuse

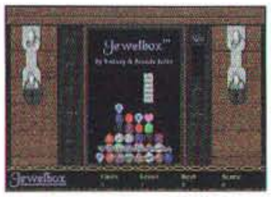
by Bungie Software

Abuse is 360 degrees of side-scrolling action. Run, jump, fall and fly in any direction - through industrial corridors, caverns and sewers. Destroy enemies in any direction with grenade launchers, rocket launchers, napalm and nova spheres! Avoid deadly traps with jet packs and turbo boost!

Key Features:

- Make your own mayhem with the Level Editor
- Hunt down your friends in 8-person multiplayer games
- Awesome Arsenal. Napalm Bombs, Nova Spheres and the Death Saber: just a few ways to lay waste!
- Point and Kill Interface. Move and annihilate mutants in complete 360° freedom
- Blast your way through floors, walls and ceilings in search of the ultimate power-up!
- Abuse is 360 degrees of side-scrolling action. Run, jump, fall and fly in any direction – through industrial corridors, caverns and sewers
- Destroy enemies in any direction with grenade launchers, rocket launchers, napalm and nova spheres! Avoid deadly traps with jet packs and turbo boost!

(SABUSE) Our Price **$51**

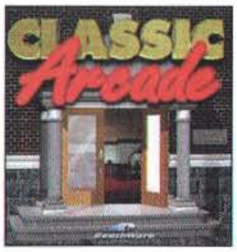


1000 Games for Macintosh

by BeachWare, Inc.

The best Macintosh game disc in the entire world, this CD-ROM contains over one thousand great shareware and public domain programs. Battle ugly aliens, blast apart run-away asteroids, deal yourself that royal flush or solve that 3-D puzzle, this disc has it all! System requirements: Mac Plus or greater, CD-ROM drive, and 2 MB of available RAM (4 MB of RAM when running under System 7).

(STGM) Our Price **$24**



Classic Arcade

by BeachWare, Inc.

Ten of your favorite coin-arcade games, redone with killer graphics and sounds! Walk through a virtual arcade and test your game playing skills with these exciting arcade classics. Ten games, including Moon Lander, Astro-Boing, Hyper Hockey, Ballistic Avenger, and more. System Requirements: Mac - Color Mac with 8 MB RAM, CD-ROM drive. PC 486 with 8 MB RAM, Sound card, SuperVGA,CD-ROM drive.

(SCLA) Our Price **$24**

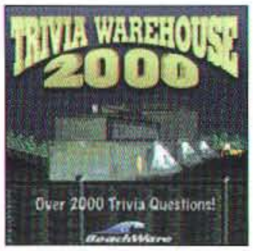
Marathon Trilogy Box Set

by Bungie Software

The Marathon Trilogy Box Set brings all three Marathon games together in one affordable package, with tons of extras thrown in. Besides Marathon, Marathon 2: Durandal and Marathon Infinity, you'll also receive a staggering 1200 maps, featuring never-released Bungie maps and the winners of the Infinity Mapmaking Contest, The Marathon Scrapbook (a behind-the-scenes look at themaking of the Marathon games), Marathon collectables like the Marathon 3-sticker set, and to top it off, the award-winning game that laid the groundwork for Marathon: Bungie's breakthrough Pathways Into Darkness.

The Marathon Trilogy Box Set is native to the Power Macintosh, utilizes the graphics acceleration of 630 and 6200 machines, is 8, 16 and 24-bit color capable, and can be played with joysticks and game pads. The package requires a 68040 or higher Macintosh,CD-ROM drive, 8-bit color monitor (13" recommended), and System 7 or later.

(SMTBS) Our Price **$65**



Trivia Warehouse 2000

by BeachWare, Inc.

Introducing a fun new PC and Mac CD-ROM. This disc contains two thousand trivia questions, in 45 categories, in several game formats! Test your memory with the Q&A, Concentration, and Multiple Choice games! Categories include: Animals, Bodies, Bond, Bugs, Cities, Comics, Geography, Gilligan, Gross, Health, Holidays, Horrors, Kids, Knot's Landing, Math, Movies, and many more. Mac System requirements: Color Mac, CD-ROM drive, 4 MB of RAM. PC system requirements: Windows 3.1 or later, CD-ROM drive, 4 MB of RAM.

(STW2K) Our Price **$24**

WAIT... There's More!

Here are more products. For full product descriptions please see our Web site, or feel free to call, fax, or E-mail us.

PRODUCT	CODE	OUR PRICE
A Zillion Sounds	SAZS	24.00
Night Sky Interactive	SNSI	24.00

System 7.5 Technologies

by Apple Computer, Inc.

- Self-paced course designed to allow software developers to write code that extends the functionality of an application for System 7.5
- Contains comprehensive materials for drag-and-drop, threads, standard mail package, and QuickDraw GX printing

Students should be familiar with the basics of developing an application on the Macintosh. Metrowerks CodeWarrior Lite is included on the CD with the lab exercises. The lab assignments were developed in CodeWarrior 8. The labs can also be done in another development environment but project files for them are not provided.

Training Format: Tutorial with labs.

Requirements: Macintosh or Mac-OS compatible computer with a 68020 processor or greater (PowerPC preferred); 8 MB RAM; 25 MB hard disk space; System 7.5 or later; CD-ROM drive.

(SSYSTECH) Our Price **$49.95**

Apple Guide Integration

by Apple Computer, Inc.

- Self-paced overview teaches you when and how to add Apple Guide help to your program.
- Powerful help system that can guide the user through a task.
- Tutorial will lead you through the steps necessary to integrate Apple Guide into your application. CodeWarrior Lite is included on the CD with the lab exercises.

Training Format: Overview with labs.

Requirements: Macintosh or Mac-OS compatible computer with 68020 processor or greater, PowerPC preferred; 8 MB RAM; 25 MB hard disk; System 7.5 or later; CD ROM drive.

(SAGI) Our Price **$49.95**

Virtual Tutor for QuickTime VR

by Apple Computer, Inc.

- Self-paced, hands-on course, which provides a comprehensive environment for learning the steps of the QTVR development process. The student can cover all of the topics or choose areas to focus on. Topics covered include: QTVR capabilities and key concepts, panoramic movies, object movies, QTVR Scene movies and authoring with QTVR
- CD-ROM contains lots of useful examples and demos. In addition to all the step-by-step exercise files

If the student completes the entire course, he/she will create a complete, authored multimedia project similar to the demonstration title that comes on the enclosed CD-ROM. There are approximately 3–4 days of training.

Training Format: Tutorial with labs.

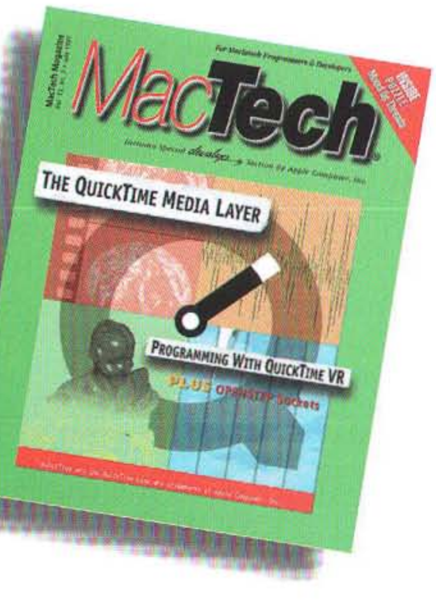
Requirements: 40 MB RAM minimum, 64 MB preferred; Macintosh or Mac OS-compatible computer with a 33 MHz 68040 processoror greater; System 7.1 or later; CD-ROM drive; 17" color monitor.

(SVTFQTVR) Our Price **$79.95**

develop

Apple's develop™ back issues are available for only $10 per copy. To place your orders, call

800/MACDEV-1.



For Macintosh Programmers & Developers

MacTech® MAGAZINE

MacTech® Magazine

MacTech keeps Mac programmers & developers up to date with everything they need to know about software development. Topics like Rhapsody, Java, QuickTime, OPENSTEP, Objective-C, C/C++, Object Oriented Technologies, product reviews and much more!

Subscriptions:

(MTYRDM) US/Domestic for 12 issues **$47**

(MTYRCM) Canadian for 12 issues **$59**

(MTYRFM) International for 12 issues **$97**

Back Issues: each plus shipping (subject to availability) **$10**

metroWerks

CodeWarrior Wear

You live it, you breath it... you might as well wear it! (XL only)

Alien T-Shirt
(AALIEN) Our Price **$9.95**

Arnold T-Shirt
(ACWARNLD) Our Price **$9.95**

Arnold Jr. T-Shirt
(AARNOLDJR) Our Price **$9.95**

Blood, Sweat & Code T-Shirt (SS)
(ACWSBLOOD) Our Price **$9.95**

Blood, Sweat & Code T-Shirt (LS)
(ACWLBLOOD) Our Price **$14.95**

CodeWarrior Baseball Cap - Black
(ACWBHAT) Our Price **$14.95**

CodeWarrior Sweatshirt - Black
(ACWSWEAT) Our Price **$29.95**

CodeWarrior Hawaii Five-0 Shirt
(ACWHAWAII) Our Price **$9.95**

CodeWarrior Winter Hat
(AWINHAT) Our Price **$14.95**

Debugger Boxer Shorts
(ADBOXER) Our Price **$16.95**

Discover Programming T-Shirt
(ADISCPT) Our Price **$9.95**

LEGTOP PODEUM™

Podeum Sport
by Rach, Inc.

Now you can have laptop stability, drop protection & mobility. If you're looking for an inexpensive, non-technical gift for a laptop owner – look no further.

- Strap your laptop to your leg
- Universal size - fits all laptops and all legs (13" - 46")
- Velcro Velcoins attach your laptop to the podeum
- Podeum allows working angles up to 90 degrees
- Dropped laptops account for 41% of all laptop fatalities
- Manufacturer's lifetime warranty.

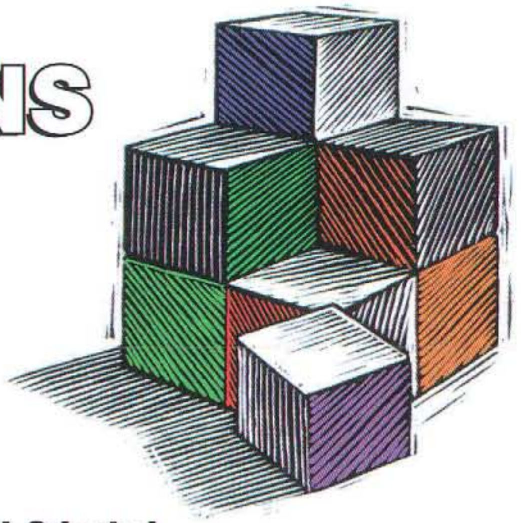
(APODSP) Our Price **$39**

MacTech® Mouse Pad

Slide on this! With an extra-large surface (11" by 10") and a deluxe sleek plastic coating, you'll be zooming across your screen in no time at all. Speed limit not enforced!

(AMTPAD) Our Price **$8.95**

NeXT PUBLICATIONS



Getting Started With WebObjects
by NeXT Software, Inc.

If you're a first time user, start here to learn the basics of how to create and run WebObjects applications.

(BGSWO) Our Price **$14**

WebObjects Developer's Guide
by NeXT Software, Inc.

A guide to building and understanding WebObjects applications. Takes a close look at the WebObjects scripting language. Additional sections explain the WebObjects architecture and tells you how to integrate your code into the request-response loop, create reusable components, create client-side components, take advantage of powerful Foundation Framework features, and more. Filled with example code. For all WebObjects programmers.

(BWODG) Our Price **$16**

D'OLE Developer's Guide
by NeXT Software, Inc.

Distributed OLE is available today, and this book shows you how to use it. Using real-world examples, the book steps you through the process of making your OpenStep objects available on Windows through OLE.

(BDOLEDG) Our Price **$22**

Discovering OPENSTEP, Mac
by NeXT Software, Inc.

Introduces programmers to NeXT's OPENSTEP 4.0 Developer product by guiding them through the creation of three applications of increasing complexity. The tutorials demonstrate and explain programming techniques, Objective-C fundamentals, common APIs, and usage of the developement tools. Along the way they present summaries of important concepts and paradigms. The book also includes a chapter directing readers to programming resources, further information, and services such as training and support. An appendix offers a concise discussion of object-oriented programming.

(BDOSTEPM) Our Price **$15**

Discovering OPENSTEP, Windows
by NeXT Software, Inc.

Discovering OPENSTEP provides an introduction to OPENSTEP programming on WIndows NT. It guides the reader through the creation of three applications of increasing complexity. Along the way, it explains concepts and illustrates aspects of Objective-C, OpenStep classes, the development environment, and programming techniques. A short appendix offers a summary of object-oriented programming.

(BDOSTEPW) Our Price **$16**

Object-Oriented Programming and Objective C
by NeXT Software, Inc.

An introduction to the principles of object-oriented programming in OPENSTEP and the official description of the Objective-C language. Objective-C is easy to learn and use because it adds very little syntax to the C programming language. It's dynamic nature allows you to accomplish things not possible in most other object-oriented languages. For any OPENSTEP programmer.

(BOOPOC) Our Price **$24**

Working w/ Interface Builder (for EOF)
by NeXT Software, Inc.

A hands-on, award-winning book designed to help you get your job done with the updated Interface Builder, released with NEXTSTEP 3.3 and the Enterprise Objects Framework 1.1. For any programmer using Interface Builder to design objects that truly work in NEXTSTEP.

(BWIB) Our Price **$24**

Using EOF 2.1 w/ OPENSTEP (Mac & Windows)
by NeXT Software, Inc.

Using Enterprise Objects Framework with OPENSTEP describes how to create an Enterprise Objects Framework application on OPENSTEP. It includes a tutorial and a chapter on creating a user interface for an OPENSTEP Enterprise Objects Framework application.

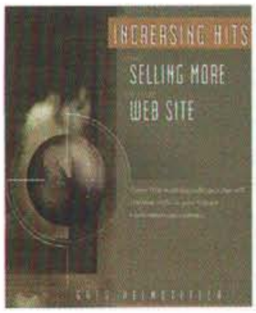
(BUEOFO) Our Price **$14**

EOF Developer's Guide for EOF 2.1(Mac & Windows)
by NeXT Software, Inc.

The Enterprise Objects Framework Developer's Guide describes how to develop database applications using the Enterprise Objects Framework tools and classes. It includes an architectural overview of the product, and descriptions of programming tips and techniques. An appendix offers a summary of Entity-Relationship Modeling.

(BEOFDG) Our Price **$24**

EOF Developer's Guide for EOF 2.0	(BEOFDG20)	Our Price **$24**
EOF Developer's Guide for EOF 1.x	(BEOFDG1X)	Our Price **$24**

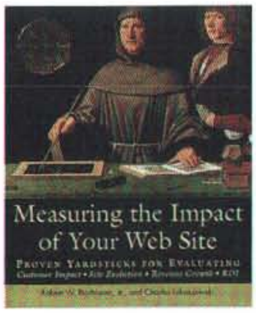


Increasing Hits and Selling More on your Web Site

by Greg Helmstetter

Written especially for entrepreneurs, corporate marketing managers, small business owners, and consultants, this valuable guide gives you rare tips and tricks you need to know to make your site a commercial success.

(BIHSMWS) Our Price **$22.45**

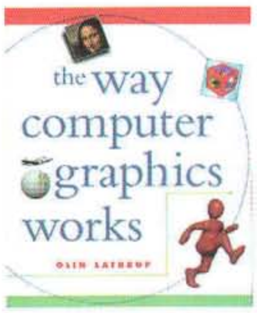


Measuring the Impact of Your Web Site

by Robert W. Buchanan and Charles Lukaszewski

This book features case studies from many successful and some less successful corporate web site pioneers. You will learn techniques and commercially available tools for measuring site traffic and visitor behavior–and help you choose the right ones for the job. Written by leading corporate site consultants this book tells you how to develop a management strategy geared toward optimizing web site productivity.

(BMIYWS) Our Price **$26.95**



The Way Computer Graphics Works

by Olin Lathrop

A complete guide to mastering computer graphic basics. It is written in a frank, down-to-earth style covering everything from how computer graphics are different from fine art and photographs, to modeling, pixels, and the principles of animation. All of this is done without resorting to mind-numbing equations and impenetrable technical jargon.

(BWCGW) Our Price **$29.65**

Debugging Macintosh Software with MacsBug

Includes CD-ROM!

by Konstantin Othmer and Jim Straus

MacsBug, from Apple Computer, Inc., is the leading debugging software program for the Macintosh. This book/disk package is an all-in-one kit for using MacsBug. Chapter 1 introduces MacsBug and describes the contents of the rest of the book. Chapter 2 describes how to install MacsBug and enough low level details about the Macintosh so that you can use MacsBug. Includes MacsBug 6.2 on disk.

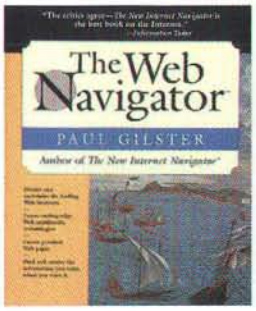
(BDMSWM) Our Price **$31.45**

The Internet Strategic Plan

by Martin A. Schulman and Rick Smith

This book gives you all of the management tools you need for creating a strong Internet and Web presence. A blueprint for your strategic plan, this book guides you every step of the way in establishing your company's place on the Internet and World Wide Web.

(BTISP) Our Price **$22.45**

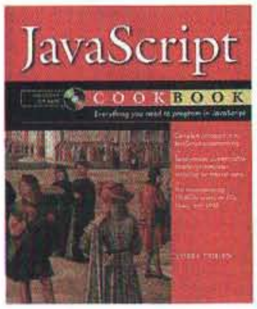


The Web Navigator

by Paul Gilster

This book keeps you right on top of all the recent changes in the Web, tells you what's out there right now and what's coming in the future. You'll get samples of various Internet sites and candid discussions of providers. You'll receive proven strategies for finding and managing information.

(BTWN) Our Price **$22.45**



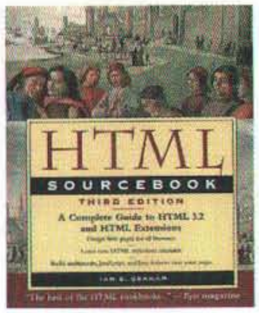
JavaScript Cookbook

Includes CD-ROM!

by Yosef Cohen

Everything you need to master the fundamentals of JavaScript programming is in this book/CD-ROM package. A special easy-to-use reference section offers detailed analysis and examples of objects, properties, event handlers, and statements. You'll also find all the latest information relevant to JavaScript programming since the advent of Navigator 2.0.

(BJSCB) Our Price **$44.99**

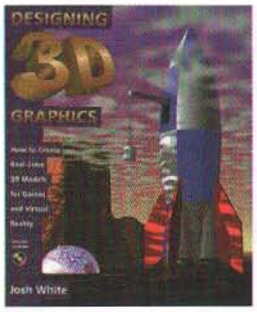


HTML Sourcebook, 3rd Edition

by Ian S. Graham

Critics everywhere agree, HTML Sourcebook is the best guide to HTML for Web professionals. That's because no other book makes it so easy for you to quickly master all the commands, tools, and expert techniques you need to create cutting-edge Web page documents. Completely revised and expanded by nearly 50 percent, this new third edition of the best-selling guide to HTML gives you the complete lowdown on all the changes and enhancements to the HTML, HTTP, and URL standards.

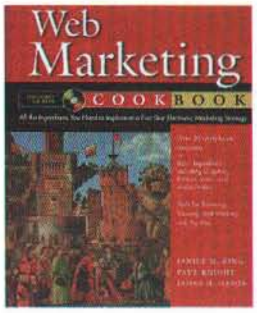
(BHTMLS) Our Price **$26.95**



Designing 3D Graphics
by Josh White

In this powerful book/CD-ROM package, top computer graphics artist Josh White tells you everything you need to know to create sophisticated real-time 3D graphics for computer games and virtual reality. This book contains the in-depth knowledge of software tools and hands-on modeling techniques that Josh White has learned while creating artwork for over 20 commercial games, including Descent, Zone Raiders, Locus, Legoland, and others.

(BD3DG) Our Price **$35.95**



Web Marketing Cookbook
by Janice M. King, Paul Knight, and James H. Mason

Create the ultimate Web marketing site, quickly and painlessly! Learn how to build a Web site for your small business or nonprofit organization with this step-by-step approach. This book/CD-ROM package contains your simple, non-technical tools for converting your print brochures, text, and graphics into a powerful online promotion.

(BWMCB) Our Price **$35.95**

Developing CGI Applications with Perl
by John Deep and Peter Holfelder

A complete step-by-step guide to creating sophisticated interactive CGI applications using Perl. If you're ready to build your own customized interactive documents, forms, graphics, and other full-feature CGI applications using Perl, then this book will show you how. Covers CGI, HTTP, and the Perl scripting language.

(BDCGIA) Our Price **$26.95**

Rhapsody DeveloperOS Guide
by Jesse Feiler

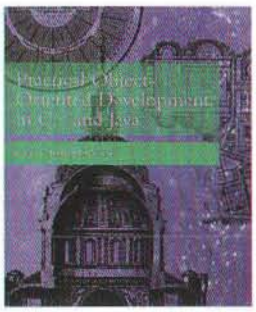
Covers the basic architectural principles of Rhapsody: the Mach microkernel, object-oriented programming, and the elements of a modern OS such as preemptive multitasking, protected memory, and symmetric multiprocessing. Also shows ways of getting to this new environment—Objective C, conversion tools, and the integration of Java—to develop Rhapsody products. Paperback, 450 pages.

(BRDG) Our Price **$35.95**

Check out our Web site!

• Full product descriptions • Hundreds of more products

http://www.devdepot.com



Practical Object-Oriented Development in C++ and Java
by Cay S. Horstmann

This book offers advice on real-world ways to use these powerful programming languages and techniques. Using the Unified Modeling Language (UML) methodology, expert Cay S. Horstmann gives you clear, concise explanations of object-oriented design, C++, and Java in a way that makes these potentially daunting operations more accessible than they've ever been before.

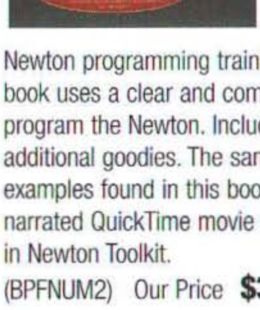
(BPOOD) Our Price **$31.50**

WebMaster in a Nutshell, Deluxe Edition
by O'Reilly & Associates, Inc.

Cross-platform, completely portable, and lightning fast, the CD-ROM is an invaluable addition to the webmaster's toolbox. The CD-ROM contains the Web Developer's Libaray—the full text of the latest editions of five popular O'Reilly titles: "HTML: The Definitive Guide, 2nd Edition"; "JavaScript: The Definitive Guide, 2nd Edition"; "CGI Programming on the World Wide Web"; "Programming Perl, 2nd Edition"; and "WebMaster in a Nutshell." The Deluxe Edition also includes a printed copy of "WebMaster in a Nutshell," the all-inclusive quick reference that belongs next to every webmaster's terminal. Includes CD-ROM & 356 page book.

Requirements: The CD-ROM is readable on all platforms, but requires a web browser that supports HTML 3.2, Java, and JavaScript.

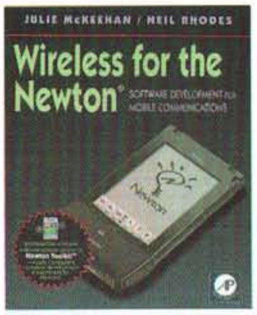
(BWMNUTD) Our Price **$62.95**



Programming For The Newton Using Macintosh, 2nd Edition
by Julie McKeehan and Neil Rhodes

This book gives you everything you need to create Newton 2.0 applications. From the people who developed the Newton programming training materials for Apple Computer, this book uses a clear and comprehensive approach to teach you how to program the Newton. Includes a CD-ROM full of sample code and additional goodies. The samples include solutions to all of the coding examples found in this book. Each example and exercise also has a narrated QuickTime movie showing the solution from start to finish in Newton Toolkit.

(BPFNUM2) Our Price **$31.45**



Wireless For The Newton

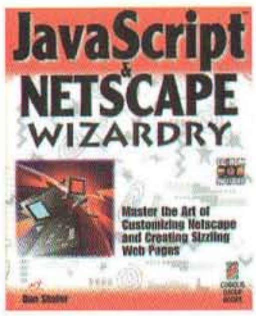
Includes Disk!

by Julie McKeehan and Neil Rhodes

A book that picks up where Programming for the Newton left off, teaching the reader how to develop Newton software on the Macintosh. The enclosed floppy disk provides a sample application, as well as a fully functional demonstration version of Newton Toolkit.

- Learn to develop Newton software on the Macintosh
- Hands-on Newton environment training with sample code
- Includes disk with sample source code for a Newton application, as well as demonstration NTK – the complete development environment for the Newton

(BWIRELESS) Our Price **$31.45**



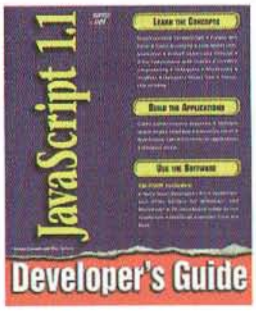
JavaScript & Netscape Wizardry

by Dan Shafer

Includes CD-ROM!

The perfect book to show you how to turn Netscape into your own personal, customized operating system. Provides the inside tips and techniques for making your Web pages much more attractive. Shows you how to use all of the key features of the JavaScript language, including objects, methods, properties, events, and much more. Includes CD-ROM with numerous interactive scripts written in JavaScript you can add to your Web pages today. A complete set of the best Java applets. Useful plug-ins designed to supercharge Netscape and resources to help JavaScript programmers.

(BJNWIZ) Our Price **$31.45**



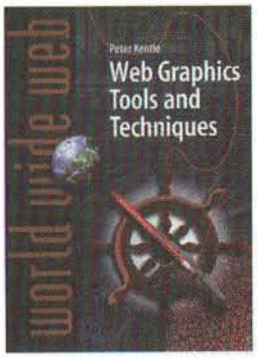
JavaScript 1.1 Developer's Guide

by Arman Danesh and Wes Tatters

Includes CD-ROM!

Written by developers for developers. An advanced guide to creating professional Web applications with JavaScript 1.1 as deployed in Netscape Navigator 3.0, Microsoft Internet Explorer 3.0, and LiveWire. Includes CD-ROM with Sun's Java Developer's Kit, JavaScript and HTML Editors for Windows and Macintosh, 20 contributed ready-to-run JavaScripts and JavaScript examples from the book.

(BJSDG) Our Price **$44.99**

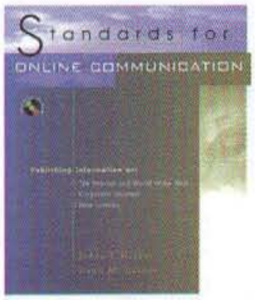


Web Graphics Tools and Techniques

by Peter Kentie

The ultimate source of information on Web graphics for both Macintosh and PC users. Recent technologies covered include: ActiveX, Sound, Adobe Acrobat, Java, VRML, QuickTime VR and video, Shockwave, and 3D Animation.

(BWGTT) Our Price **$35.95**



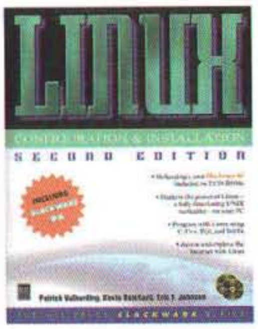
Standards For Online Communication

by JoAnn T. Hackos and Dawn M. Stevens

Includes CD-ROM!

Guidelines for how tooplace information online within your company. Provides both a design and development process and a set of guidelines for the Internet, intranets, and help systems for designers and authors who need to create effective electionic information. Includes CD-ROM with software containing files to help you utilize the models described in the book.

(BSFOC) Our Price **$40.45**



Linux Configuration & Installation, 2nd Edition

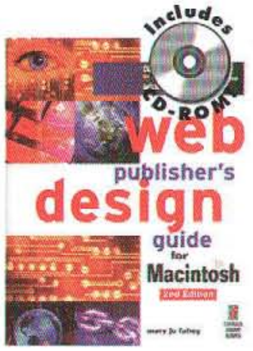
by Patrick Volkering, Kevin Reichard, and Eric F. Johnson

Includes CD-ROM!

Linux, the leading UNIX variant, has garnered loads of attention within the UNIX community. The amazing thing about Linux is that you don't need a workstation to run it. Linux Configuration & Installation, Second Edition lets you run Linux today. Program with Linux using C, C++, Perl, and Tcl/Tk. The 2 CD-ROM pack offers one of the most popular Linux distributions, Slackware 96, and comes directly from Patrick Volkering, the creator of Slackware.

(BLCI2) Our Price **$35.95**

One & 2-Day SHIPPING (call for details)



Web Publisher's Design Guide for Macintosh, 2nd Edition

by Mary Jo Fahey

This is the only book that takes you step-by-step through real projects designed by talented new media artists. Internet design experts share their design secrets and art files (look for art files on the companion CD-ROM by artist's last name). This expanded new edition includes Photoshop, Illustrator and DeBabelizer tricks from the first edition plus countless new ways to liven up your Web pages with 3D graphics, sound, movies, and much more.

(BWPDG2) Our Price **$35.99**

BASIC for the Newton

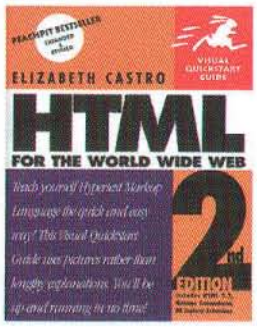
by John Schettino and Liz O'Hara

Includes Disk!

Program on Macintosh, Windows-based PC, or on the Newton itself. Straight-forward "programming by example" approach – you'll be writing Newton programs right away. Includes 3.5" disk containing Demonstration NS BASIC and over fifty example programs (Newton not included).

(BNEWT) Our Price **$32.35**

SEE RELATED CATEGORY: Dev. Environments

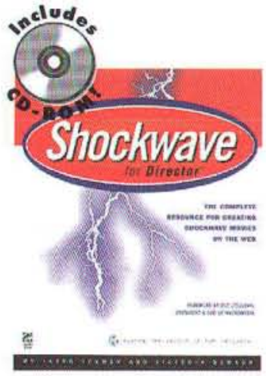


HTML For The World Wide Web, 2nd Edition

by Elizabeth Castro

Teach yourself Hypertext Markup Language the quick and easy way! This Visual QuickStart Guide uses pictures rather than lengthy explanations. You'll be up and running in no time. If you need to learn HTML fast – this is book is for you.

(BHTMLW2) Our Price **$16.15**

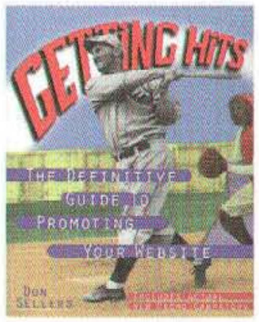


Macromedia Shockwave for Director

by Jason Yeaman and Victoria Dawson

The complete resource for creating Shockwave movies on the Web. This hands-on reference makes it easy to create Shockwave movies and put them on the Web. Expert tips from the creators of Macromedia's first Shockwave movies, together with detailed examples and instruction, provide everything you need to get started. Includes CD-ROM.

(BMSFD) Our Price **$27**

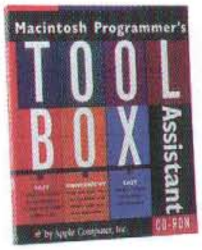


Getting Hits–The Definitive Guide To Promoting Your Website

by Don Sellers

Getting Hits explains in easy-to-understand language the underlying concepts behind the art of Web site promotion. Just a few of the topics you'll learn include: using search engines with URL's; finding related Internet groups or lists; understanding the nuances of click throughs and ad rates; and distributing press releases to key Internet contacts. With this book, you'll go beyond the conceptual and actually follow real-world tested promotional campaign strategies. Bring the world to your Web site!

(BGHITS) Our Price **$17.95**



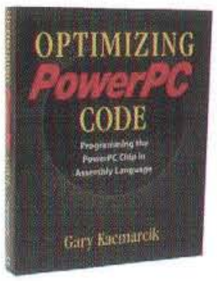
Programmer's Toolbox Assistant CD-ROM

Instant electronic access to Inside Macintosh essentials.

by Addison-Wesley Publishing

Get quick access to reference pages for over 4,000 Toolbox calls in your system software from their development environment. Essential information for Macintosh software developers. Hypertext links allow programmers to view related topics easily. The ultimate electronic reference tool for Macintosh programmers.

(STBASST) Our Price **$89.95**

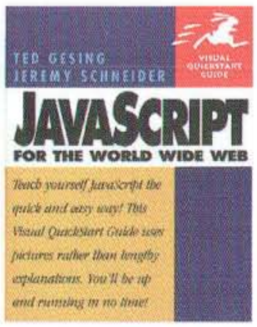


Optimizing PowerPC Code: Programming the PowerPC in Assembly Language

by Gary Kacmarcik

Take full advantage of the potential of the PowerPC by mastering the Assembly Language techniques. Learn to produce faster more robust software!

(BOPTPPC) Our Price **$35.96**

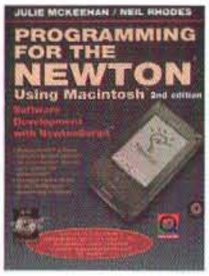


JavaScript For The World Wide Web

by Ted Gesing and Jeremy Schneider

This book takes an easy, visual approach to teaching JavaScript, where pictures guide you through the software and show you what to do. Works like a reference book, you look up what you need and then get straight to work. No long winding passages, concise, straightforward commentary explains what you need to know.

(BJWWW) Our Price **$16.15**

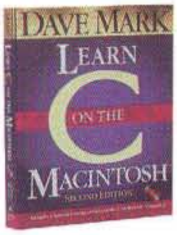


Programming for the Newton Using Macintosh:Software Development with NewtonScript- Second Edition

by Julie McKeehan & Neil Rhodes

Praise for the Second Edition! "Rewritten from cover to cover, this new edition teaches you the essentials of programming for Newton 2.0. You must read this book. Then read it again. And again . . ."--SCRIBBLES (OxNUG, Australian Newton Users Group) Includes one CD-ROM for Macintosh 68030 or higher. 466 pp.

(BPFNUM) Our Price **$31.45**



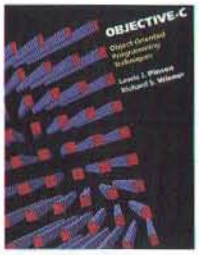
Learn C on The Macintosh Second Edition

by Dave Mark

New revised edition! Easy-to-understand – everything you need to start programming. Updated and enhanced exercises that lead you step by step. You'll learn function, variables, point datatypes, data structures, file input and output and more! Includes CD-ROM with Metrowerks CodeWarrior™ Lite.

(BLEARNC2) Our Price **$33.25**

SEE RELATED CATEGORY: Dev. Environments



OBJECTIVE-C Object-Oriented Programming Techniques

by Lewis J. Pinson and Richard S. Wiener

Presents the basic concepts of object-oriented design and programming, and provides a precise description of the Objective-C language. Several small-to-medium sized applications using Objective-C illustrate the general principles of object-oriented programming. Covers the two main versions of the Objective-C language. Demonstrates the versatility and power of the NeXT machine as a platform to support object-oriented programming. Shows how to design, implement, and use hierarchies of classes. Explains the purpose of pre-defined classes and shows how they can be used in designing programs.

(BOBJCOOPT) Our Price **$34.15**

Inside CodeWarrior Professional

by Metrowerks

Includes CodeWarrior IDE User's Guide. This is the printed version of the documentation provided on the CD. Covers CodeWarrior Professional Release, the debugger and associated tools.

(BINSCWP) Our Price **$34.95**

SEE RELATED CATEGORY: Dev. Environment

Check out our Web site!

• Full product descriptions • Hundreds of more products

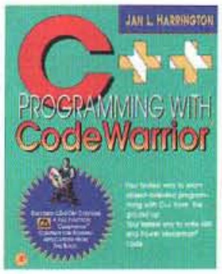
http://www.devdepot.com

Metrowerks CodeWarrior Programming

by Dan Parks Sydow

Includes CodeWarrior Lite, and Full Coverage of PowerPlant™. The best information on Metrowerks CodeWarrior, giving full coverage to the Gold Edition. CD includes Code Warrior Lite.

(BCWPROG) Our Price **$35.95**



C++ Programming with CodeWarrior

by Jan L. Harrington

Beginning OOP for the Macintosh and Power Macintosh and Mac OS compatibles. Learn object-oriented programming techniques using C++ as the example language and Metrowerks and CodeWarrior as the example compiler. Enclosed CD contains example code from the book and a full-function Metrowerks CodeWarrior.

(BCPPCW) Our Price **$32.35**

CodeWarrior Software DevelopmentUsing PowerPlant

by Jan L. Harrington

C++ programmers will learn to develop object-oriented software applications for the Mac and Power Mac using the PowerPlant environment and the classes that support it. Covers CodeWarrior 8. Included CD-ROM contains source code for all the programming examples in the book and Metrowerks CodeWarrior Lite.

(BCWSWDEV) Our Price **$31.45**

Inside PowerPlant

by Metrowerks

Create PowerPlant applications using the CodeWarrior IDE and PowerPlant Constructor. Full descriptions of major PowerPlant classes and resources. Included are the PowerPlant Constructor Manual, including View, TextTraits and Custom Types editing, and PowerPlant Library Reference, covering all classes and functions in PowerPlant.

(BINSPP) Our Price **$34.95**

SEE RELATED CATEGORY: Dev. Environment

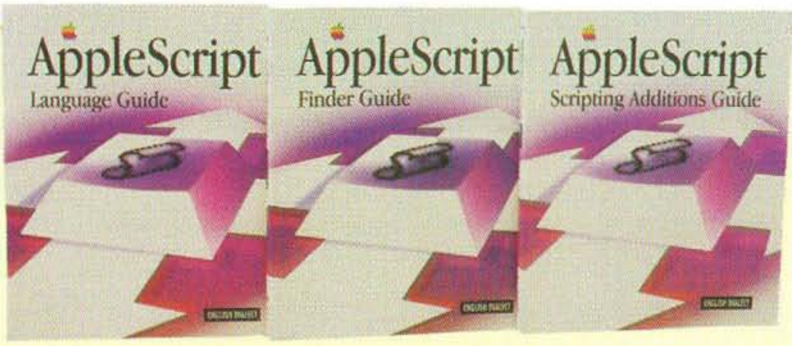
AppleScript Finder Guide, English Dialect

by Apple Computer, Inc.

Provides definitions for Finder object classes and commands. Write, record, or run scripts that trigger the same desktop actions that you trigger using the keyboard and mouse.

(BAFG) Our Price **$17.95**

SEE RELATED CATEGORY: Scripting



AppleScript Language Guide
AppleScript Finder Guide
AppleScript Scripting Additions Guide

AppleScript Language Guide

by Apple Computer, Inc.

A complete reference for anyone using AppleScript to modify existing scripts or to write new ones. Contains useful information for programmers who are working on scriptable applications or complex scripts. Features detailed definitions of AppleScript terminology and syntax in the following categories: Value classes, commands, objects and references to objects, expressions, control statements, handlers, and script objects. Includes many sample scripts, discusses advanced topics such as writing command handlers for script applications, the scope of script variables and properties declared at different levels in a script, and inheritance and delegation among script objects.

(BALG) Our Price **$26.95**

SEE RELATED CATEGORY: Scripting

AppleScript Applications: Building Applications with FaceSpan and AppleScript

Includes CD-ROM!

by John Schettino Affiliation & Liz O'Hara

Build complete AppleScript applications using FaceSpan, a user interface development tool that makes AppleScript applications truly "Mac-Like". Uses a step-by-step approach to demonstrate techniques for building applications through illustrations and samples. Provides Graphical User Interface (GUI) design tips and practical approaches for implementation. Contains one CD-Rom with AppleScript 1.1, a demonstrations version of FaceSpan 2.1, source code for all example applications numerous AppleScript shareware and demonstrations programs. Contains a section on debugging AppleScript applications using FaceSpan.

(BAPSCAP) Our Price **$31.45**

Special Edition Using CGI, 2nd Edition

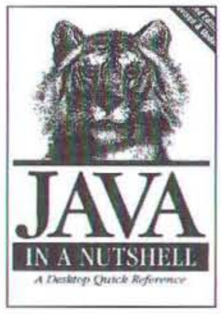
Includes CD-ROM!

by Jeffry Dwight, Michael Erwin and Robert Niles

This complete reference provides professional Web developers and advanced personal users with the latest information on using CGI (Common Gateway Interface) to interact with databases.

- Explains client and server uses of CGI
- Provides extensive coverage of live audio and video feeds, user chat and interaction, and CGI security
- Features separate chapters devoted to language-specific tips, tricks, and traps
- CD ROM is loaded with the HTML and CGI sample code from the book
- Includes applications for guest books, mail and new gateways, browser identification, access restriction, and shopping carts

(BSEUCGI) Our Price **$44.99**



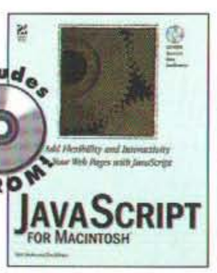
Java in a Nutshell, 2nd Edition

by David Flanagan

A detailed overview of all of the new features in Java 1.1, both on a package-by-package basis and in terms of overall functionality. A comprehensive tutorial on "inner classes" that explains how to use all of the new types of inner classes: static member classes, member classes, local classes, and anonymous classes. Practical, real-world example programs that demonstrate the new features in Java 1.1, including object serialization, the new AWT event handling model, internationalization, and a sample Java Bean.

(BJNUT2) Our Price **$17.95**

Lowest price guaranteed. Feel Good about your purchase! 30-day return policy.



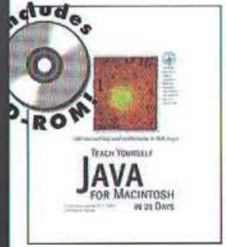
JavaScript for the Macintosh

Includes CD-ROM!

by Matt Shobe and Tim Ritchey

Allows non-programmers to take advantage of the power of Netscape Navigator. Expand the capabilities of your Web page, without having to understand C or C++. CD-ROM contains "Wizlets" that allows you to easily create your own JavaScripts. Takes you step-by-step through programming cross-platform JavaScripts. Details how to create JavaScripts for JavaScript-aware Web browsers.

(BJAVASCRPTJ) Our Price **$40.50**

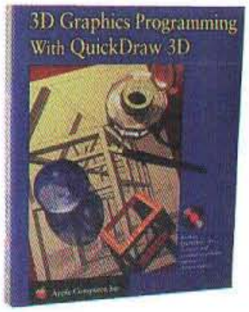


Teach Yourself Java for Macintosh in 21 Days

by Laura Lemay and Charles L. Perkins with Timothy Webster

Add interactivity and multimedia to Web pages! A step-by-step guide to make your Website come alive. Learn the basics of programming Java applets and the concepts behind the Java language. Includes CD-ROM with a limited version of Roaster, the first commercial, integrated applet development environment for Java for the Macintosh!

(BJAVAMAC) Our Price **$36**

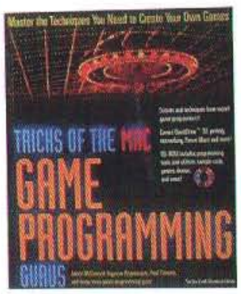


3D Graphics Programming Using QuickDraw 3D

by Apple Computer, Inc.

Incorporate spectacular 3D graphics into your applications. Explore QuickDraw 3D, a revolutionary graphics extension to the Mac OS for Power Macintoshes. CD contains the complete QuickDraw 3D system itself and a complete database of the QuickDraw 3D API, allowing you instant access to the hundreds of graphics calls via a fast viewing engine. Book/CD-ROM, 640 pages.

(B3DGRAP) Our Price **$35.96**

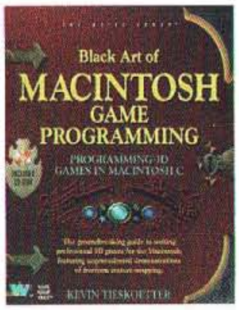


Tricks of The Mac Game Programming Gurus

by McCornack, Ragnemalm, Celestin, et al.

For beginning to expert game programmers. Complete overview of all the necessary components of game programming on the Macintosh. Packed with valuable tools, utilities, sample code, CodeWarrior™ Lite and game demos. QuickDraw 3D and Power Mac optimization and inside info on how Glypha III was created. Hundreds of tried-and-true tricks, tips, and insider secrets from well-known Mac game programming experts.

(BTRICKS) Our Price **$45**



Black Art of Macintosh Game Programming

by Kevin Tieskoetter

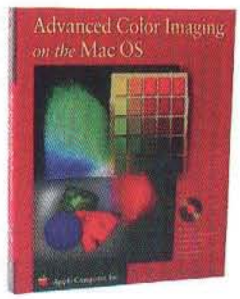
Develop your own 3D games in C on the Mac. Includes CD with project files for both Symantec C and Code Warrior. Create freeform texture-mapped games and polygon graphics. Control dynamic source code - all compatible as native to the Power Mac. Write directly to the screen, bypassing QuickDraw.

(BBLACK) Our Price **$35.99**

Check out our Web site!

• Full product descriptions • Hundreds of more products

http://www.devdepot.com

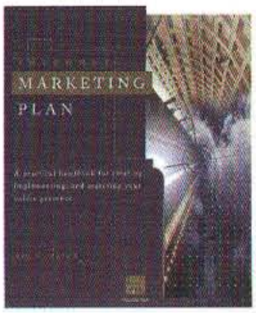


Advanced Color Imaging on the Mac OS

by Apple Computer, Inc.

Enhance your software's color capabilities with step-by-step instructions. Augment the color support supplied with QuickDraw, and QuickDraw GX. Use the Pallette Manager to get the best colors on limited displays. Match colors between screens and input/output devices (scanners & printers). CD includes a complete reference information in both QuickView and Acrobat formats. Plus, a sample application demonstrating ColorSync programming techniques.

(BADVCI) Our Price **$33.25**

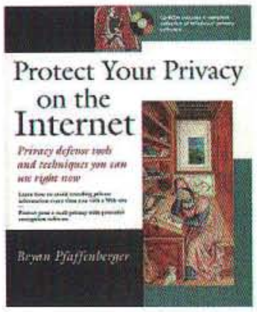


The Internet Marketing Plan

by Kim M. Bayne

This book gives you a comprehensive framework for producing and executing a customized Internet marketing plan. Marketing communications veteran Kim Bayne supplies you with a clear set of step-by-step procedures for establishing, implementing, evaluating, and managing your company's online presence.

(BTIMP) Our Price **$35.99**



Protect Your Privacy on the Internet

by Bryan Pfaffenberger

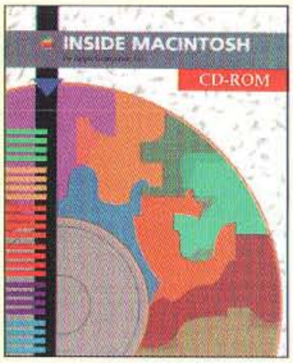
This book/CD-ROM package gives you proven privacy defense strategies and techniques to help you make the Net a safer place to work and play. You'll get the names of Internet privacy organizations that are working to protect your privacy rights and find out what you can do to help.

(BPYP) Our Price **$26.99**

Order Toll-free
800-MACDEV-1
(800-622-3381)

INSIDE MACINTOSH

by Apple Computer, Inc.



Inside Macintosh: CD-ROM

by Apple Computer, Inc.

More than 25 volumes in electronic form. Includes: QuickDraw™ GX Library, Macintosh Human Interface Guidelines, PowerPC System Software, Macintosh Toolbox Essentials and More Macintosh Toolbox, QuickTime and QuickTime Components. Access over 16,000 pages of information with Hypertext linking and extensive cross referencing.

(BIMCD) Our Price **$89**

WAIT... There's More!

Limited time offer to buy these Inside Macintosh products – 10 % off! For full product descriptions please see our Web site, or feel free to call, fax, or E-mail us.

PRODUCT	CODE	OUR PRICE
Inside Macintosh: Devices	BIMDEV	26.95
Inside Macintosh: Files	BIMFIL	26.95
Inside Macintosh: Interapplication Communications	BIMIAPP	33.25
Inside Macintosh: Memory	BIMMEM	22.45
Inside Macintosh: More Macintosh Toolbox	BIMMAC	31.45
Inside Macintosh: Networking	BIMNET	26.95
Inside Macintosh: Overview	BIMOVER	22.45
Inside Macintosh: PowerPC Numerics	BIMPPCNUM	26.05
Inside Macintosh: PowerPC System Software	BIMPPCSYS	22.45
Inside Macintosh: Processes	BIMPROC	20.65
Inside Macintosh: QuickDraw GX Prog. Overview	BIMGXOV	22.45
Inside Macintosh: QuickDraw GX Typography	BIMGXTYP	26.95
Inside Macintosh: QuickTime	BIMQT	26.95
Inside Macintosh: QuickTime Components	BIMQTCOM	31.45
Inside Macintosh: Sound	BIMSOUND	26.95
Inside Macintosh: X-Reference	BIMXREF	17.95

(Book sale prices are contingent upon availability)

For Macintosh Programmers & Developers

MacTech®
M A G A Z I N E

MacTech® Magazine

MacTech keeps Mac programmers & developers up to date with everything they neec know about software development. Topics like Rhapsody, Java, QuickTime, OPENSTEI Objective-C, C/C++, Object Oriented Technologies, product reviews and much more!

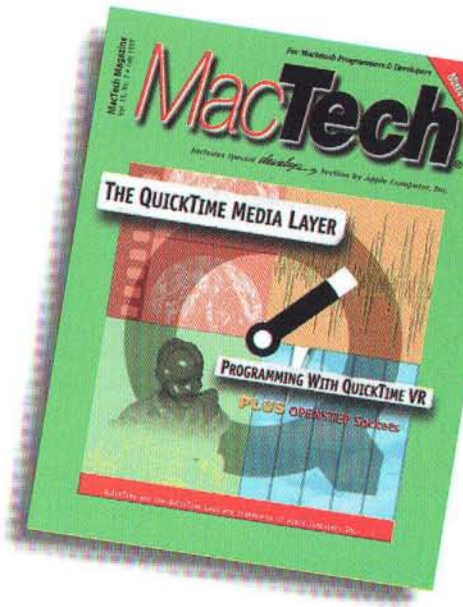
Subscriptions:

(MTYRDM) US/Domestic for 12 issues **$47**

(MTYRCM) Canadian for 12 issues **$59**

(MTYRFM) International for 12 issues **$97**

Back Issues: each plus shipping (subject to availability) **$10**



MacTech® CD-ROM Volumes 1-12

- Includes Apple's *develop* issues 1-29 (1990-1997)
- Almost 1600 articles from all 139 issues of MacTech Magazine (1984-1996) and through may of 1997
- Improved hypertext, improved indices, and a new THINK Reference Viewer – for lightning quick access!
- New hyperlinks between articles
- 100+ MB of source code — use them in your applications, with no royalties!
- Full version of THINK Reference™ — the original online guide to Inside Macintosh, Vols. I-VI
- 80MB of FrameWorks/SFA archives and the most complete set of FrameWorks archives known
- Sprocket™! MacTech's tiny framework that compiles quickly and supports System 7.5 features
- The best threads from the Macintosh programmer newsgroups plus thousands of notes, tips, snippets, and gotchas
- Popular tools that Macintosh programmers use to increase their productivity and much more!

(SMTCD12) Volumes 1-12 Our Price **$129**

(SMTCD12U) Upgrade from any previous version Our Price **$49**

WAIT... There's More!

Here are more products. For full product descriptions please see our Web site, or feel free to call, fax, or E-mail us. Our prices on books are at least 10% off list price.

PRODUCT	CODE	PRICE
Development Environments		
AppleScript Applications: Building Applications w/FaceSpan	BAPSCAP	$31.45
Basic for the Newton - Programming using NS BASIC	BNEWT	32.35
C++ Programming	BCPPMACAP	31.46
C++ Programming with CodeWarrior	BCPPCW	32.35
C/C++ SDK User's Guide	BCPPUSER	29.00
CodeWarrior Inside PowerPlant	BINSPP	34.95
CodeWarrior Software Development using PowerPlant	BCWSWDEV	31.45
Cyberdog Programmers Kit	BCYBERDOG	34.95
Dan Shafer Presents the Power of Prograph CPX	BDANPRO	19.95
Inside CodeWarrior Book	BINSCW	34.95
Instant CORBA	BIC	17.99
Last Resort Programmers Edition	BLSTRSRT	74.95
Learn C on the Macintosh, 2nd Edition	BLEARNC2	33.25
Learn C++ on the Macintosh	BLRNCPP	35.05

PRODUCT	CODE	PRICE
Macintosh C Programming Primer Volume I	BCPRIM1	24.25
Macintosh C Programming Primer Volume II	BCPRIM2	24.25
Macintosh Pascal Programming Primer Volume I	BPASCPRI	24.25
Mastering the Think Class Library	BMASTERTCL	26.95
Mastering the Toolbox Using THINK C	BCPRIM2	24.25
Metrowerks CodeWarrior Programming	BCWPROG	35.95
Objective-C - Object-Oriented Programming	BOBJCOOPT	34.15
Presenting Magic Cap	BPRESMAGIC	15.25
Real World Apple Guide	BREALWLD	35.95
Symantec C++ Programming	BSYMCPP	39.50
Taligent's Guide to Designing Programs	BTALIGENT	17.55
The Power of Prograph CPX	BDANPRO	19.95
Visual Programming with Prograph CPX	BVISPRO	30.60
Wireless For The Newton	BWIRELESS	31.45

Hardware

Apple CD-ROM Handbook	BCDHAND	14.36
Designing Cards & Drivers for the Macintosh	BCARD	26.96
LaserWriter Reference	BLASERREF	17.96
PCI System Architecture 3rd Edition	BPCISYS	31.46
PowerPC System Architecture	BPPCARCH	31.46

Internet Related

1994 Internet White Pages	B94WHITE	26.95
Active Java	BACTJAVA	23.36
America Online for Dummies	BAOLDUM	17.95
CGI By Example	BCGIBE	31.45
Building & Maintaining an Intranet w/ the Macintosh	BBAMAI	45.00
Claris Home Page Companion	BCHPC	26.95
Computer Privacy Handbook	BPRIV	22.45
E-Mail Essentials	BEMAILE	22.45
Elements of E-Mail Style	BEMAIL	13.45
Hooked on Java	BHJAVA	26.95
Instant Internet Guide	BINSTANT	13.45
Internet Book	BTHENET	22.50
Internet for Dummies 2nd Edition	BNETDUM2	17.99
Internet for Dummies Quick Reference	BDUMQCK	8.05
Internet for Macs for Dummies	BNETDUM	17.95
Internet for Macs for Dummies Bestseller Edition	BIFMFDBE	35.99
Internet Power Tools	BPWRTOOL	36.00
Internet Publishing with Adobe Acrobat	BIPWAA	36.00
Internet, The, Deluxe Edition	BNETDELUX	31.50
Intranet Web Dev.: Enterprise Alternatives to Client/Server	BINTWD	44.99
Java Essentials for C/C++ Programmers	BJAVAESSEN	17.95
Java in a Nutshell	BJAVANUT	13.45
Java Language API SuperBible	BJLAS	53.99
Java Programming with CORBA	BJPWC	26.99
JavaScript for Macintosh	BJAVASCRPT	40.50
Learn HTML on the Macintosh	BLHTML	26.95
Learn Java on the Macintosh	BLJAVA	31.45
Mastering Netscape 2.0 for Macintosh, Second Edition	BMASNET2	36.00
More Internet for Dummies Starter Kit	BDUMNET	17.95
Mosaic for Dummies	BMOSDUM	17.99
Net Chat	BNETCHAT	17.00
NetObjects Fusion Handbook	BNETOFH	45.00
Netscape Navigator 3.0	BNETN3	26.96
Netscape Navigator Starter Kit for Macintosh	BNETNSK	31.49
Perl Quick Reference	BPERLREF	17.99
Planning and Managing Websites	BPLANWEB	35.96
Protect Your Privacy on the Internet	BPYP	26.99
Providing Internet Services via the MacOS	BPROVNET	31.46
Publish it on the Web	BWEBPUB	31.46
TCP/IP Vol 1-Vol 2 Bundle	BTCP12BNDL	99.00
Teach Yourself Java in 21 days	BJAVAMAC	36.00
The Internet Marketing Plan	BTIMP	35.99
Underground Guide to Telecommuting	BUNDER	22.45
Using Lotus Notes as an Intranet	BULN	40.45
Web Head Mac Guide	BWEBHEAD	22.45
Web Page Scripting Techniques	BWEBPST	45.00
Web Publishing with Adobe Acrobat and PDF	BWPWAA	35.95
Web Weaving	BWWEAV	22.45
Webmaster Macintosh	BWEBMAS	26.95

Scripting and Solutions

AppleScript Applications: Building Apps with FaceSpan	BAPSCAP	31.45
AppleScript Scripting Additions Guide	BSCRADD	17.05
Applied Macintosh Scripting	BAPPLIED	31.45
Complete HyperCard 2.2 Handbook	BHYPCRD2	31.50
Complete HyperTalk 2.2:	BHYPCRD2	31.50
Danny Goodman's Apple Guide Starter Kit	BDGAGSK	31.46
HyperCard Stack Design	BHYPSTA	19.95
JavaScript for Macintosh	BJAVASCRPT	40.50
Perl Quick Reference	BPERLREF	17.99
Real World Apple Guide	BREALWLD	35.95

Technical Reference

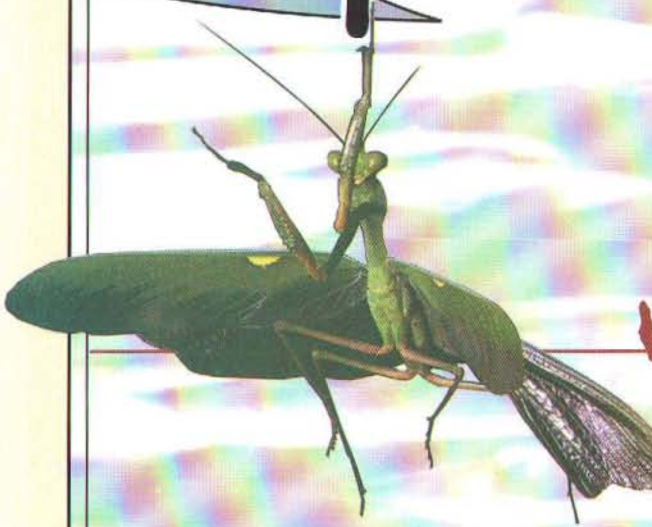
Active Java	BACTJAVA	23.36
Apple CD-ROM Handbook	BCDHAND	14.36
Art of Human Interface Design	BAHID	29.65
Black Art of Mac Game Programming	BBLACK	35.99
C++ for Dummies	BCPPDUM	22.46
C for Dummies Vol. 1	BCDUM	17.95
Developing Object Oriented Software for the Macintosh	BDEVOB	26.06
Extending the Mac Toolbox	BETMT	22.46
Foundations of Macintosh Programming	BFOUND	35.96
Fragment of Your Imagination	BFRAG	35.96
Guide to Macintosh Software Localization	BLOCALIZ	24.26
Guide to Macintosh System 7.5	BSYS7.5	22.50
Inside AppleTalk	BAPTALK	31.45
Inside the Macintosh Communications Toolbox	BCOMM	22.45
LaserWriter Reference	BLASERREF	17.96
Learn C++ on the Macintosh	BLRNCPP	35.05
Learn C on the Macintosh, 1st Edition	BLEARNC1	31.45
Learn C on the Macintosh, 2nd Edition	BLEARNC2	33.25
Mac Programming for Dummies	BMACDUM	17.95
Macintosh C Programmer Primer Volume 1	BCPRIM1	24.25
Macintosh C Programmer Primer Volume 2	BCPRIM2	24.25
Macintosh OLE2 Prog. Reference Working with Objects	BOLE2	40.45
Macintosh Pascal Programming Primer Volume I	BPASCPRI	24.25
Macintosh Programming Secrets 2nd edition	BPSECRET	28.76
Macintosh Programming Techniques	BPTECH	31.95
Microsoft Visual J++ 1.1 Sourcebook	BMVJS	35.95
More Mac Programming Techniques	BMORETECH	34.50
Network Frontiers Bundle	BNETFB	59.95
Newton Programming Guide	BNEWTPGUID	40.46
Object Oriented Programming Design	BOOPRODES	20.66
Optimizing PowerPC Code	BOPTPPC	35.96
Perl Quick Reference	BPERLREF	17.99
PostScript Language Reference	BPSLANREF	29.66
Powerbook: Digital Nomad's Guide	BPBTDNG	22.46
PowerPC Programmer's Toolkit	SPPCPT	40.50
Programming Introduction to the Macintosh Family	BFAMILY	22.46
Programming for System 7	BSYS7	24.25
Programming Primer Macintosh Volume 1	BPRIMMAC	34.15
Programming Starter Kit	BPROSTART	40.50
Programming with AppleTalk	BPROAT	22.45
QuickTime - Official Guide for Macintosh Users	BQTGUIDE	45.00
Real World Apple Guide	BREALWLD	35.95
ResEdit All Night Diner	BRESDINE	22.45
ResEdit Complete, 2nd Edition	BRESED2	31.45
ResEdit Reference	BRESEDREF	26.96
Software by Design: Creating User Friendly Software	BDESIGN	26.95
Symantec C++ for the Macintosh: The Basics	BSCFTMTB	31.46
Teach Yourself Macintosh C++ in 21 Days	BCPP21D	26.99
Technical Introduction to the Macintosh Family	BTITTMF	24.26
Tog on Software Design	BTOG	26.95
Wireless for the Newton Development for Mobil Comm.	BWIRELESS	31.45
Writing Localizable Software	BLOCAL	24.25

Miscellaneous

3D Graphics-Tips, Tricks, & Techniques	B3DGTTT	31.46
A Fragment of Your Imagination	BFRAG	35.96
Adobe Premiere for the Macintosh	BPREM	44.95
America Online for Dummies	BAOLDUM	17.95
AppleGuide Complete	BAPLGD	35.96
Art of Human Interface Design	BAHID	29.65
CD-ROM Guide to Multimedia Authoring	BCDMULTI	40.45
CompuServe for Dummies	BCSDUM	17.95
Creating Interactive CD-ROM	BINTERCDR	35.95
Cyberpunk Handbook	BCYBPUNK	8.95
Danny Goodman's Apple Guide Starter Kit	BDGAGSK	31.46
Danny Goodman's Macintosh Handbook	BGOODHB	26.95
FrameWorks Source Code Disk	MTFWSC	9.95
FrameWorks Magazine Back Issue	MTFWBACK	8.00
Global Interface Design	BGLOBAL	32.35
Graphic Gems 2	BGEMS2	44.95
Graphic Gems 4	BGEMS4	44.95
Graphic Gems V	BGEMS5	44.95
Infini-D Revealed	BINFDREV	40.50
Inside Director 5 with Lingo for Macintosh	BID5WLM	44.99
Late Night with MacHack	BLATE	26.95
Mac Bathroom Reader	BBATH	11.70
Macromedia Director Lingo Workshop, 2nd Edition	BMDLW2	40.50
Mac Screamer: The Ultimate Macintosh Supercharging Kit	BSCREAM	31.50
Macintosh Crash Course	BCRASH	26.95
MacTech Back Issues	MTBACKISS	10.00
Macworld Ultimate Macintosh Programming Book	BULTMAC	35.95
Managing AppleShare & Workgroup Servers	BMAWS	26.95
MADACON '93 CD-ROM	SMADA93	9.95
Multimedia Authoring: Building and Developing Documents	BMMAUTH	31.45
Multimedia Starter Kit for Macintosh	BMMSTART	27.00
Network Frontiers Bundle	BNETFB	59.95
Profit from Experience	BPROFIT	22.45
ResEdit Complete, Second Edition	BRESED2	31.45
Sad Macs, Bombs and Disasters	BSADMAC	22.45
Stupid Mac Tricks	BSTUPIDMAC	17.95
The Elements of E-Mail Style	BEMAIL	13.45
The Software Developer's & Marketer's Legal Companion	BSDAMLC	33.26
Tog on Software Design	BTOG	26.95
Tricks of the Mac Game Gurus	BTRICKS	45.00
Zen and the Art of Resource Editing	BZAAORE	27.00

All entries in this index are alphabetized. For your convenience the product names are **bold**, book names are *italicized* and company names are in plain text.

purity
software, inc.



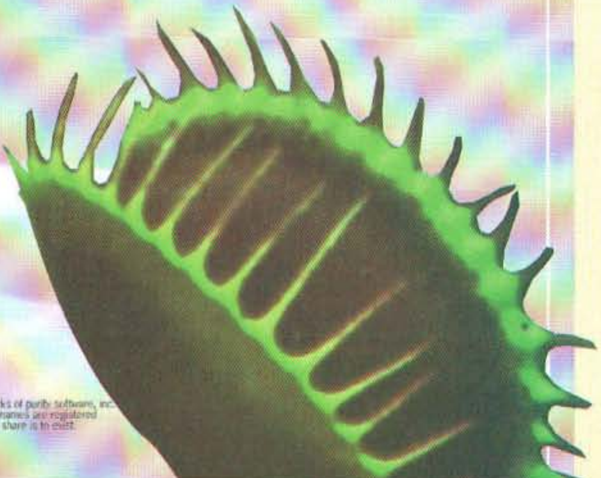
websiphon™
$495

WebSiphon allows you to rapidly design custom server-side web applications by embedding an easy to learn, yet powerful scripting language directly into your HTML documents. WebSiphon also includes Verona, the only Macintosh flat-file database server written specifically for use on a web site. If you have ever felt restricted by other web site development products on the market, then this is the tool for you.

- PowerPC™ Native
- Multi-threaded
- Compiled Language
- Caching

webSentinel™
$99

WebSentinel replaces or supplements your web server's built-in security, allowing more flexibility and power to manage users and secure areas with a complete Macintosh user interface. Features include GREP match strings, custom "No Access"™ files for each realm, and full caching. WebSentinel's plug-in architecture allows you to integrate with your existing user data and security architectures.



demos

www.purity.com
1016 mopac circle, suite 101
austin tx 78746
phone 512.328.2288
fax 512.328.2688
info@purity.com

CODEWARRIOR LATITUDE

KICKING BUTT AND WRITING CODE

CodeWarrior Latitude

When Rhapsody gets here, you'll be ready...with CodeWarrior Latitude™. It's Metrowerks' newest porting tool, designed to give you a leg up on Rhapsody. Recompile your Mac OS source code, link it with the Latitude libraries and find out which portions of your code are going to port smoothly...and which won't [forewarned is forearmed]. As Rhapsody evolves, so will Latitude; registered users will receive all developer releases, the first full release, plus one additional update. CodeWarrior Latitude. $399. The tools you need...and a little attitude to boot.

COOL TOOLS FOR KILLER CODE.

metrowerks®

http://www.metrowerks.com